

BASIC JUNIOR

INICIACION A LA PROGRAMACION

RICARDO AGUADO-MUÑOZ
AGUSTIN BLANCO
ENRIQUE RUBIALES
JAVIER ZABALA
RICARDO ZAMARREÑO

De los mismos autores:

BASIC BASICO
CURSO DE PROGRAMACION

PROGRAMAS COMENTADOS DE
BASIC BASICO

BASIC JUNIOR

INICIACION A LA PROGRAMACION

RICARDO AGUADO-MUÑOZ
AGUSTIN BLANCO
ENRIQUE RUBIALES
JAVIER ZABALA
RICARDO ZAMARREÑO

JAVIER ZABALA
Vitrubio, 3
28006 MADRID
Tfno: 91-262 49 25

RICARDO ZAMARREÑO
Bristol, 10 - 9.° izda.
28028 MADRID
Tfno: 91-256 23 67

DISTRIBUYE:
GRUPO DISTRIBUIDOR EDITORIAL, S.A.
Don Ramón de la Cruz, 67
28001 MADRID
Tfno: 91-401 12 00

Editan:

I.S.B.N.: 84-398-2592-7
Depósito legal: M. 36.938-1984
Imprime: Hijos de E. Minuesa, S. L. Ronda de Toledo, 24 - 28005 Madrid.

INDICE

PRESENTACION

El uso de los *microordenadores* se ha extendido rápidamente entre los *más jóvenes;* sin embargo, hay pocos libros en castellano destinados específicamente a ellos. BASIC JUNIOR *Iniciación a la Programación* pretende contribuir modestamente a llenar esa laguna.

Entre las aplicaciones más interesantes de los ordenadores figuran, sin duda, los *juegos;* pero el juego más fascinante que puede jugar un chaval es la *programación* de los juegos por él imaginados. Los capítulos de este libro son como las piezas de un mecano: aislados valen poco, pero hábilmente combinados pueden dar origen a bellas y divertidas creaciones.

Otro aspecto interesante para un chico es la utilización del ordenador como un *instrumento para aprender.* Y no nos referimos a la aburrida E.A.O. (Enseñanza Asistida por Ordenador), sino al uso del *micro* como una herramienta más de su aprendizaje, con la que puede crear pequeños ficheros, hacer un diccionario personal de idiomas, preparar un trabajo escrito o descomponer un número en factores primos.

Los microordenadores modernos, además de la facilidad que tienen para hacer cálculos y escribir textos, ofrecen amplias *posibilidades gráficas.* Por ello hemos creído conveniente tratar muchos de los temas desde este punto de vista.

Hemos intentado emplear un lenguaje claro y asequible, en el que ha primado, más que el rigor técnico, la intención didáctica. El libro está dividido en veinticinco capítulos cortos; al final de cada uno de ellos van cuatro ejercicios cuidadosamente seleccionados, que el lector debe intentar hacer. En los casos de mayor dificultad damos alguna idea para resolverlos.

Una vez más hemos tenido la suerte de poder contar con las ilus-

traciones de nuestro amigo *Saltés*, que contribuyen no sólo a arrancar la sonrisa del lector, sino a explicar lo que con palabras no queda claro.

Esperamos que BASIC JUNIOR tenga la misma acogida que sus hermanos mayores BASIC BASICO *Curso de programación* y *Programas comentados de* BASIC BASICO, libros a los que remitimos a quienes quieran seguir profundizando en este lenguaje.

LOS AUTORES

1

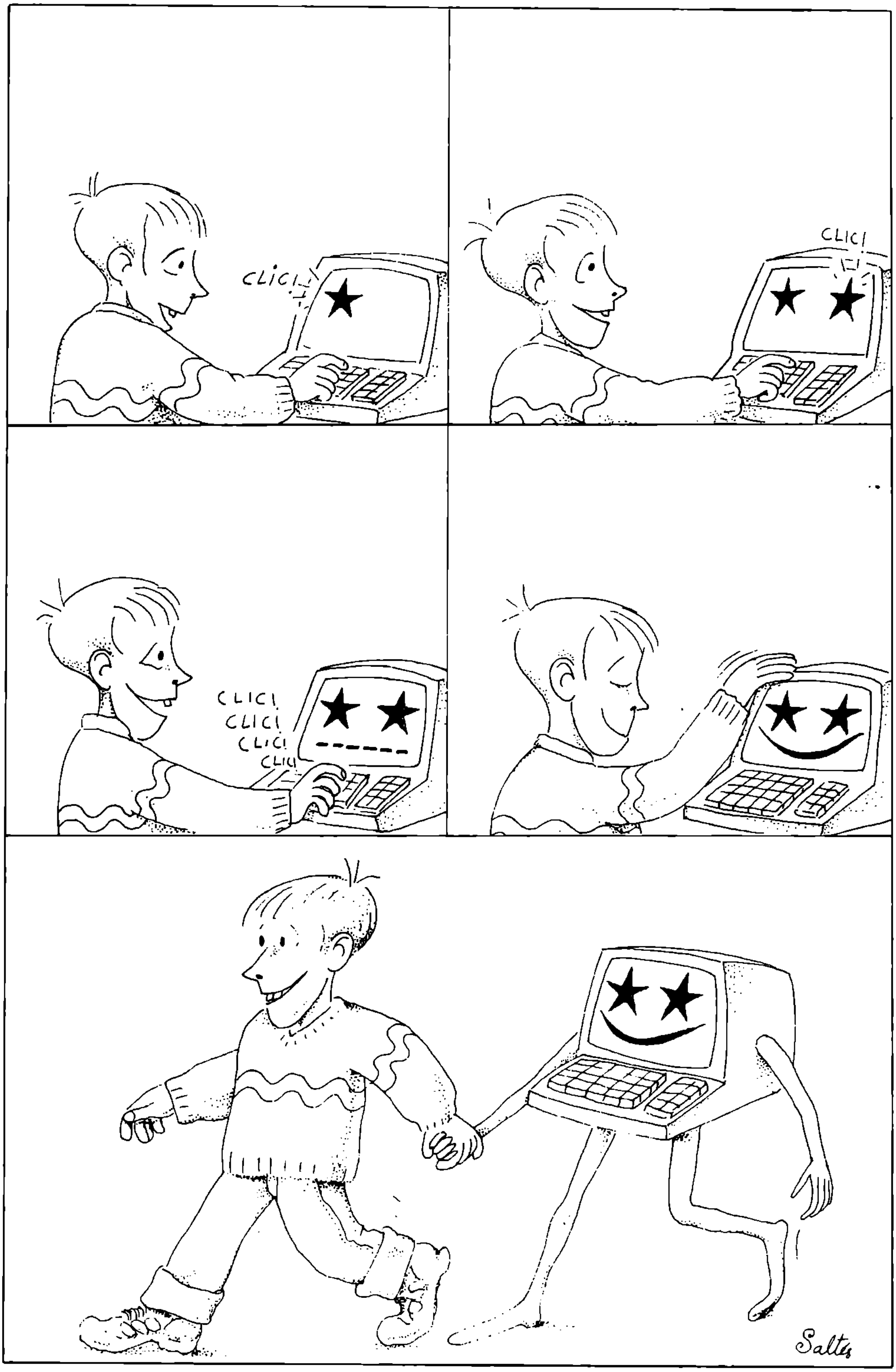
EMPEZANDO A PROGRAMAR

1.1 INTRODUCCION

Algunos creen que la Informática y los ordenadores son cosa de magia; que es muy fácil con ellos acertar una quiniela de catorce resultados, predecir el tiempo que va a hacer el próximo verano o enviar un satélite a la Luna. Pero están muy equivocados.

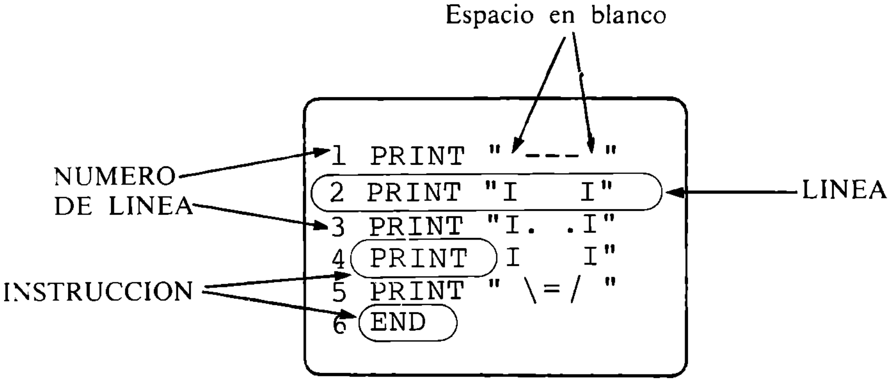
Un ordenador no hace nada si no se le pide correctamente. La forma de pedirle algo, de decirle lo que queremos, se llama *programa.* Lo que es sorprendente es la velocidad y exactitud con que ejecuta los programas. Es como si tuviera dentro un ejército de pequeños robots trabajando bien y a toda velocidad. Pero los robots necesitan un hombre que les dirija, que les diga qué hay que hacer y cómo. Y ese hombre has de ser tú.

En el mundo hay muchos idiomas que se componen de miles y miles de palabras. Para programar los ordenadores se han inventado unos idiomas llamados *lenguajes de programación.* Están muy bien pensados y con 30 ó 40 palabras en inglés puedes hacer ya muchas cosas. A las faltas de ortografía se les llama *errores de sintaxis* (el ordenador no perdona ni uno). Aquí vamos a estudiar el lenguaje BASIC, que es el más empleado, y el mejor para los principiantes.


CLIC!
CLIC!
CLIC!
CLIC!
CLIC!
CLIC!
Salts

1.2 TU PRIMER PROGRAMA

Vamos a dibujar la caricatura de Robotín, que es nuestro amigo dentro del ejército de robots que trabaja en el ordenador. Teclea lo que sigue, pulsando al final de cada línea la tecla RETURN (si tu ordenador no tiene esta tecla, pulsa la tecla ENTER , o bien la NEW LINE).



PROGRAMA

Esto es un programa que se compone de seis *líneas.* Cada línea empieza por un *número de línea.* Luego viene una palabra que es PRINT en las primeras líneas y END en la última. Estas palabras se llaman *instrucciones.*

La instrucción PRINT ordena imprimir (sacar en pantalla). En este programa lo que está entre las comillas será dibujado en la pantalla. Imagínate a Robotín pintando su autorretrato. Ten cuidado en respetar los espacios en blanco.

La instrucción END indica el fin del programa.

Los números de línea sirven para indicar a la máquina en qué orden debe obedecer las instrucciones. No importa escribir las líneas salteadas, porque el ordenador las ejecuta por orden de números de menor a mayor.

Para ejecutar el programa tienes que escribir la palabra RUN y

pulsar [RETURN] . Inmediatamente aparecerá en la pantalla la caricatura de Robotín.

En este primer programa hemos numerado las líneas de uno en uno, pero en realidad no suele hacerse así. Lo que haremos frecuentemente será numerarlas de 10 en 10.

Vamos a programar nuevamente la caricatura de Robotín numerando las líneas de 10 en 10. Pero... ¡atención!, antes tienes que borrar de la memoria el programa anterior para que no se mezcle con el nuevo.

Para borrar el programa anterior escribe la palabra NEW y pulsa la tecla [RETURN]. Ahora ya puedes escribir:

```
10 PRINT " --- "
20 PRINT "I   I"
30 PRINT "I. .I"
40 PRINT "I   I"
50 PRINT " \=/ "
60 END
```

¿Qué ventajas tiene el numerar de 10 en 10? Que se pueden intercalar nuevas líneas. Por ejemplo, si queremos alargar la cara de Robotín, podemos intercalar entre las líneas 40 y 50 la línea:

```
45 PRINT "I   I"
```

Para intercalar esta línea, bastará que la escribas y pulses [RETURN]. Corre el nuevo programa (RUN [RETURN]) y verás que a Robotín se le ha puesto la "cara larga".

Nota: [RETURN] se mete dentro de un rectángulo para indicar que es una tecla, y que no vale escribirlo letra a letra.

1.3 EN RESUMEN

A estas alturas ya has comprobado que para pasar de una línea a la siguiente hay que dar a la tecla [RETURN] o [ENTER] o [NEW LINE],

como se llame en tu ordenador. Además de pasar a la línea siguiente, esta tecla da por terminada la línea en que estabas y la mete en memoria. De ahí sus posibles nombres: RETURN =volver, ENTER=introducir y NEW LINE=línea nueva.

Los ordenadores suelen tener en la pantalla un cuadradito que parpadea, indicando dónde saldrá la próxima tecla que des. Se llama *cursor*. Al pulsar RETURN el cursor pasa al principio de la siguiente línea.

Para ejecutar el programa tienes que teclear el comando RUN (y pulsar RETURN , naturalmente). RUN significa correr. Prueba a ejecutar varias veces el programa que tengas en la memoria. Robotín se pone a trabajar (en este programa a dibujar) a toda velocidad, de línea en línea.

EJERCICIOS

1.1 Prepara un programa que dibuje este pez:

```
    * * * *
  * *       *   *
*   *       * *
  * *       *   *
    * * * *
```

1.2 Prepara un programa que dibuje la casa:

```
      *
    *   *
  *       *
* * * * * * *
*           *
*           *
*           *
*           *
* * * * * * *
```

1.3 Haz un programa que dibuje la cara de este señor:

```
/ / / / / /

!         !
! ( . . ) !
!         !
!   , ,   !
!   - -   !
 V V V V V
```

1.4 Haz cuatro programas para dibujar las figuras:

```
* * * * *   *             * * * * *   * * * * *
*       *   * *             * * * *   * * * * *
*       *   * * *             * * *   * * * * *
*       *   * * * *             * *   * * * * *
* * * * *   * * * * *             *   * * * * *
```

En cada programa tienes que utilizar 5 instrucciones PRINT (no te olvides de las comillas, ni de incluir los espacios que sean necesarios para dibujar las figuras).

2

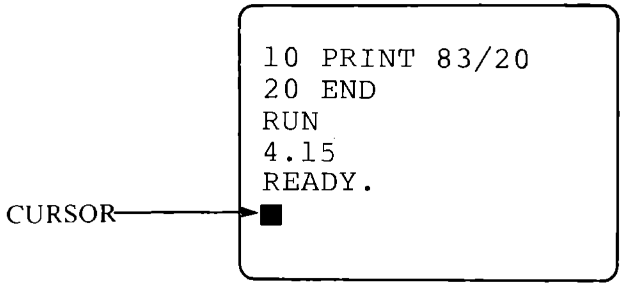
NUEVAS INSTRUCCIONES Y COMANDOS

2.1 EL TABACO

Robotín odia el tabaco, que además es perjudicial para sus circuitos. Cansado de todos los argumentos médicos, él quiere aportar el argumento económico: fumar es carísimo. ¿Cuánto cuesta un pitillo? Una cajetilla de tabaco cuesta 83 ptas, y como tiene 20 pitillos, cada uno sale a...

```
10 PRINT 83/20
20 END
```

En adelante procuraremos numerar las líneas de 10 en 10. Al ejecutar este programa, en la pantalla resulta 4,15. Aquí la instrucción PRINT no lleva comillas porque no queremos dibujos, sino cálculos.



Puedes comprobar que en BASIC el signo de la división es la barra inclinada (/) y no el signo (÷), ni los dos puntos (:). ¡No te confundas! Habrás observado además que, para señalar los decimales, se pone un punto (.) y no una coma (,). Así es la notación anglosajona. Entonces los números como mil o un millón no llevan ningún tipo de puntuación: 1000, 1000000.

Si una persona fuma una cajetilla diaria, al mes se gasta...

```
10 PRINT 83*30
20 END
```

Al rodar el programa, en la pantalla sale 2490 (sin ningún punto). Fumar es un verdadero lujo. En este programa ves que el signo de la multiplicación es el asterisco (*) y no el aspa (×), ni el punto (·).

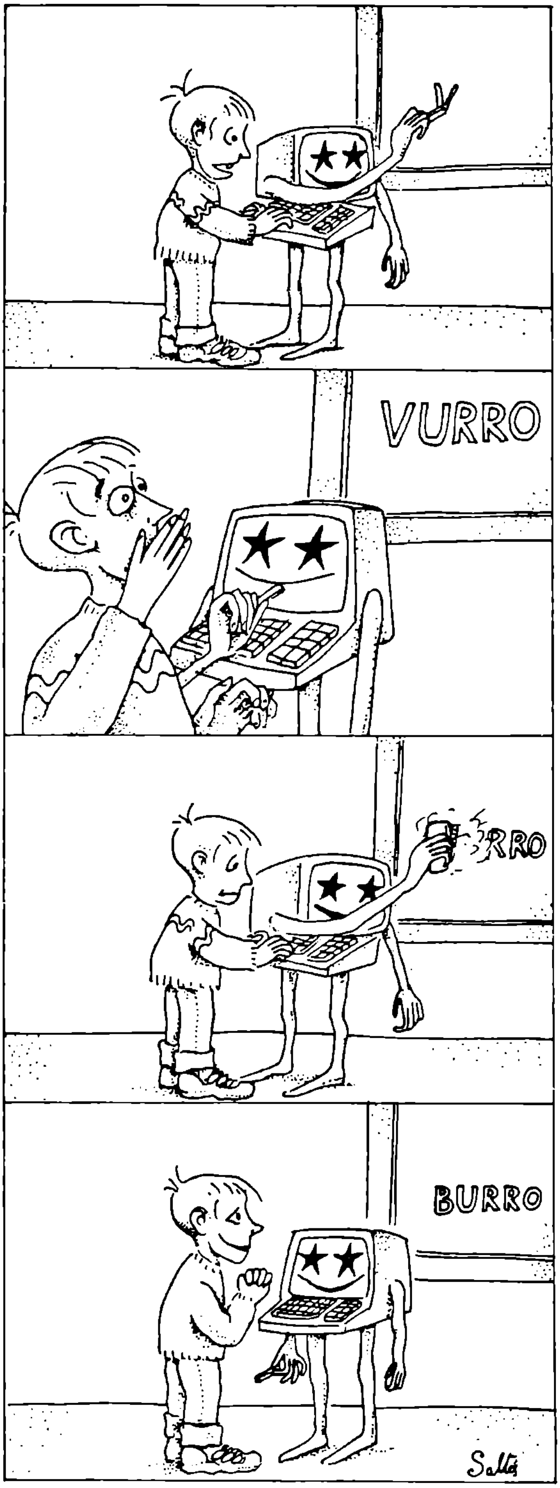
```
10 PRINT 83*30
20 END
RUN
2490
READY.
■
```

2.2 CORREGIR UNA LINEA

Hay muchas marcas de tabaco. Podemos repetir los cálculos para distintos precios. Si la cajetilla cuesta 105 ptas, en vez de 83, habrá que rectificar el programa anterior escribiendo:

```
10 PRINT 105*30
```

Cuando pulsemos la tecla RETURN, la máquina acepta esta nueva versión de la línea 10, olvidando la antigua.


VURRO
RRO
BURRO
Salto

Este sistema, consistente en escribir de nuevo una línea, sirve también para corregir los errores de sintaxis. Por ejemplo:

ERROR DE SINTAXIS → línea 10 (PRNT)

CORRECCION DEL ERROR → segunda línea 10

```
10 PRNT 105*30
20 END
RUN
? SYNTAX ERROR IN LINE 10
10 PRINT 105*30 [RETURN]
RUN
3150
READY.
■
```

Este método se utiliza también para borrar líneas. En efecto, si escribes:

10 [RETURN]

la nueva versión de la línea 10 es "nada", así que la has borrado.

Para borrar un programa podríamos borrar una por una todas las líneas, pero en programas largos sería muy pesado. Ya sabes que lo más sencillo es utilizar el comando NEW, que borra todas las líneas de golpe. Se llama *comando*, como RUN, para distinguirlo de las instrucciones, ya que los comandos van sin número de línea y no forman parte de los programas.

Si después de NEW escribes RUN, el ordenador contesta READY. y termina: no tiene ningún programa que rodar.

2.3 COMANDO LIST

Sucede a veces que el programa ya no se encuentra en la pantalla, porque la has borrado, o porque a medida que has escrito cosas nuevas

han desaparecido por arriba las antiguas. Si necesitas ver el programa para estudiarlo o corregirlo, puedes conseguir que vuelva a la pantalla escribiendo LIST y pulsando RETURN . En algunos ordenadores el comando LIST suele admitir complementos así:

`LIST 20`	lista la línea 20
`LIST -50`	lista todas las líneas hasta la 50
`LIST 30-`	lista desde la línea 30 hasta el final
`LIST 30-50`	lista las líneas 30 a 50

Algunas marcas de microordenadores responden de manera diferente. Unos, en vez de utilizar el guión (-), utilizan la coma (,). En otros el comando LIST 20 lista todas las líneas a partir de la 20.

Si después de NEW tecleas LIST, el ordenador contesta READY. Es lógico, porque con el programa borrado no tienen nada que listar.

```
NEW
READY.
RUN
READY.
LIST
READY.
■
```

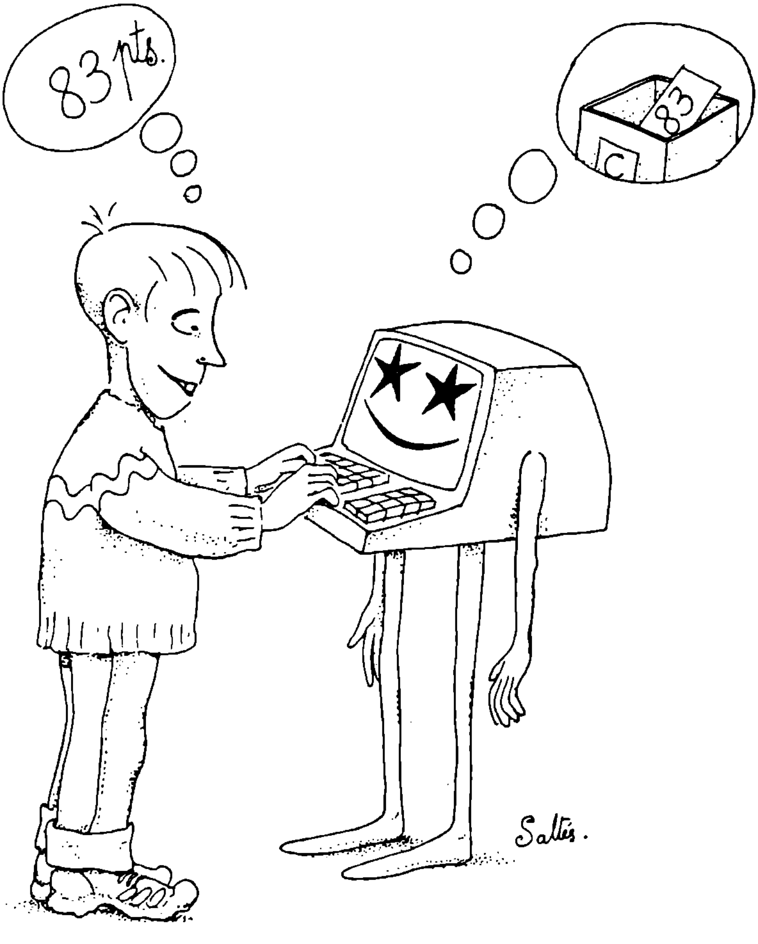
2.4 LA INSTRUCCION LET

Volvamos a tratar de los fumadores, del precio de los pitillos y del gasto mensual cuando se cambia de marca de tabaco. Ya hemos visto que un procedimiento es teclear otra vez la línea 10 con el nuevo precio:

```
10 PRINT 105*30
```

Si la línea 10 es larga, es pesado corregirla, y más, si ese dato (105) se utiliza en varias líneas. Viene en nuestro auxilio la instrucción LET. Mira el siguiente programa:

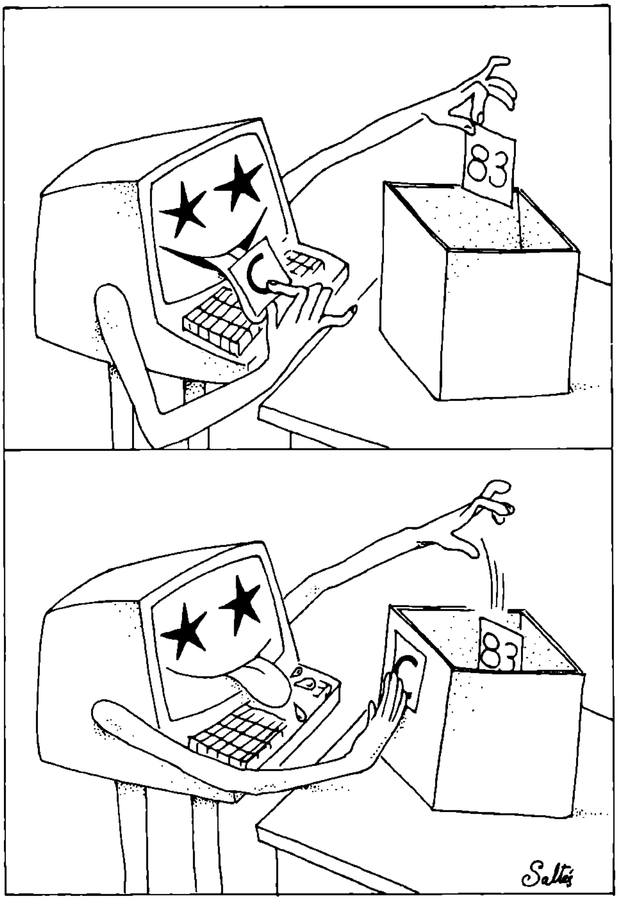
```
10 LET C=83
20 PRINT C/20
30 END
```



La línea 10 es una *asignación:* LET C = 83 significa: sea C = 83. La letra C es una *variable*, y 83 es el valor que se guarda en ella. La memoria del ordenador funciona como un gran casillero con miles de casillas. Cada vez que Robotín se encuentra con una instrucción como ésta:

```
LET C=83
```

busca una casilla vacía, le pega la etiqueta C y mete dentro el número 83.



En un programa más largo, donde la variable C se utilice en muchos lugares, se agradece el poder cambiar su valor modificando una sola línea. Prueba el siguiente ejemplo en tu ordenador:

```
10 PRINT "PRECIO DE LA CAJETILLA"
20 LET C=83
30 PRINT C
40 PRINT "PRECIO DEL PITILLO"
50 PRINT C/20
60 PRINT "GASTO MENSUAL"
70 PRINT C*30
80 END
```

Para ver la ruina que supone cambiarse a un tabaco más caro, basta escribir:

```
20 LET C=105
```

y, tras pasar el programa, estudiar los resultados sin alterar ninguna otra línea. Pruébalo.

2.5 INPUT

Esta es la instrucción apropiada para asignar con agilidad valores a una variable, sin tener que andar corrigiendo líneas de programa. Veámoslo con el ejemplo anterior:

```
10 PRINT "PRECIO DE LA CAJETILLA"
20 INPUT C
30 PRINT "PRECIO DEL PITILLO"
40 PRINT C/20
50 PRINT "GASTO MENSUAL"
60 PRINT C*30
70 END
```

Cuando, al ejecutar el programa, el ordenador llegue a la línea 20, sacará una interrogación en pantalla y se parará. Robotín ha elegido una casilla y le ha puesto el nombre C, y se queda esperando a que le demos el valor que queramos. Al escribir 83, o cualquier otro valor, y pulsar la tecla [RETURN], la máquina mete ese valor en la casilla C y continúa la ejecución del programa.



Sale una interrogación y se para — Escribimos la respuesta — Al dar a [RETURN] continúa el programa

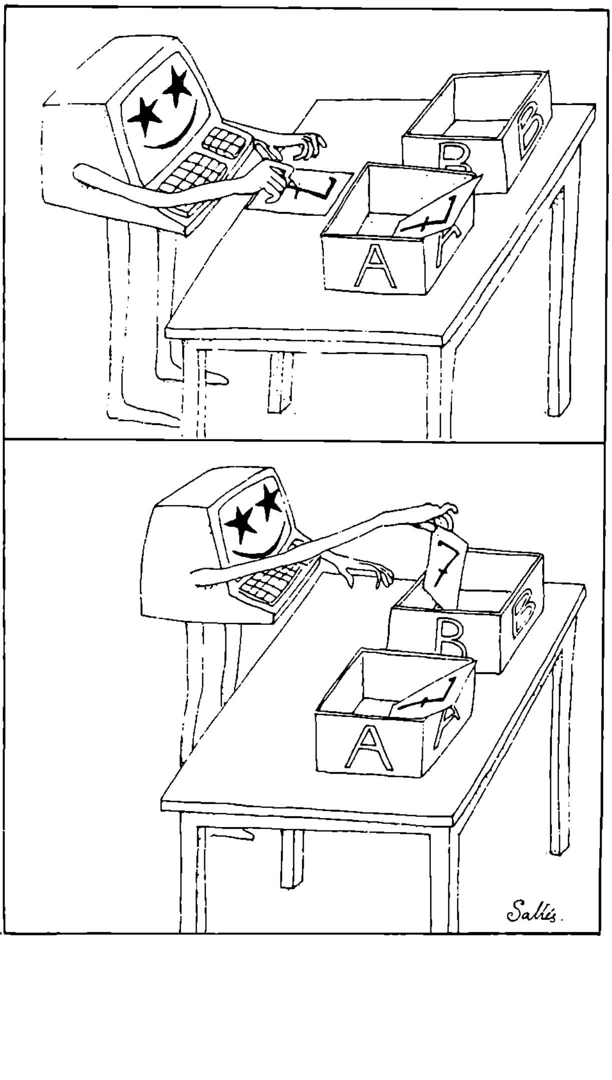
Entenderás mejor todo esto si lo practicas algunas veces. Hay algún ordenador en que el INPUT, en vez de sacar una interrogación, presenta otro tipo de señal. Si el tuyo es de ésos, convendrá que lo mires en el manual de instrucciones.

2.6 ASIGNACION

La instrucción LET no es tan fácil como parece. Mira las siguientes líneas:

```
10 LET A=7
20 LET B=A
30 PRINT B
40 END
```

La línea 10 ya la entiendes, es un LET en el que el nombre de la cajita es A. Pero ¿qué hace la línea 20? Esta línea ordena a Robotín elegir una nueva cajita y llamarle B. Después coge una copia del contenido de la cajita A y la mete en B.



La instrucción LET primero calcula lo que hay a la derecha del signo igual (=) y después lo mete en la cajita de la izquierda. Así que LET no es una igualdad, sino una asignación. Recuerda: primero se calcula la expresión de la derecha y después se mete en la variable de la izquierda.

Un último ejemplo para que lo acabes de entender:

```
10 LET A=7
20 LET A=A*2
30 PRINT A
40 END
```

¿Qué pasa con la línea 20? ¿Es un error que merece un "cate"? Pues no. Vamos a acompañar a Robotín línea a línea para que nos lo explique.

— En la línea 10 da el nombre de A a una cajita y mete dentro el valor 7.

— En la línea 20 saca una copia de A, que vale 7, la multiplica por 2, y guarda su valor en la misma cajita A. El nuevo valor, 14, anula el antiguo. Antes A valía 7; ahora vale 14.

— En la línea 30 se imprime A, esto es 14, y en la 40 se termina el programa.

2.7 VARIABLES

Ya hemos mencionado varias veces esta palabra. Se ha comenzado a explicar en la pregunta anterior. Una *variable* es una cajita de la memoria del ordenador, conocida por un nombre, y que contiene un número.

Procura llamar a las variables por nombres fáciles de recordar. Al precio de una Cajetilla le hemos llamado C. El precio del Pitillo podría ser P por ejemplo, y el gasto Mensual del fumador, M. Tratar de coger la inicial de la palabra puede ser un buen criterio. Escribe y corre este programa:

```
10 PRINT "PRECIO DE LA CAJETILLA"
20 INPUT C
30 PRINT "PRECIO DEL PITILLO"
40 LET P=C/20
50 PRINT P
60 PRINT "GASTO MENSUAL"
70 LET M=C*30
80 PRINT M
90 END
```

EJERCICIOS

2.1 Intenta predecir lo que tiene que salir en la pantalla con estos programas. Pásalos por tu "micro" y compara los resultados con tus predicciones:

```
10 LET A=5          10 LET A=3            10 PRINT 3+7
20 PRINT "A"        20 LET B=2            20 PRINT "3+7"
30 PRINT A          30 PRINT "A+B="       30 END
40 END              40 PRINT A+B
                    50 END
```

2.2 ¿Qué hace el ordenador con este programa?

```
10 LET N=0
20 LET N=N+1
30 PRINT N
40 LET N=N+1
50 PRINT N
60 LET N=N+1
70 PRINT N
80 END
```

2.3 Prepara un programa para que el ordenador te pida (INPUT) el radio R de una circunferencia y a continuación escriba (PRINT) la longitud L. (L= 2×3,14×R).

2.4 En el micromercado hacen un microdescuento de un 3,5 % sobre el valor de la compra efectuada. Haz un programa para que el ordenador tome (INPUT) el precio normal y escriba (PRINT) el precio reducido.

3

PALABRAS

3.1 CADENAS

En el capítulo anterior hemos manejado números y operaciones, y cuando nos ha interesado los hemos guardado en *variables numéricas.* Recuerda líneas como las siguientes:

```
20 LET C=83
50 PRINT C/20
70 LET M=C*30
```

Hemos manejado también palabras o frases cortas, que se ponían entre comillas:

```
10 PRINT "PRECIO DE LA CAJETILLA"
60 PRINT "GASTO MENSUAL"
```

Y, anteriormente, poníamos entre comillas las partes de un dibujo. Todos estos signos entre comillas reciben el nombre de *cadenas de caracteres* (el espacio en blanco es un carácter), para indicar que nó son números. Mira los siguientes ejemplos de cadenas:

```
"BASIC JUNIOR"
"VOLKSWAGEN"
"27 DE MARZO DE 1922"
```

La diferencia esencial es que con los números se pueden hacer operaciones, mientras que parece absurdo operar con cadenas. Las cadenas se caracterizan por ir entre comillas.

Compara las salidas de los dos programas siguientes:

```
10 LET A=18
20 PRINT A/3
30 END
RUN
6
READY.
■
```

```
10 LET A=18
20 PRINT "A/3"
30 END
RUN
A/3
READY.
■
```

Has comprobado que lo contenido entre comillas se imprime en la pantalla tal como estaba, y desde luego sin operar. Rueda en tu ordenador los siguientes programas comprobando los resultados:

```
10 PRINT "2+3"
20 PRINT 2+3
30 END
RUN
2+3
5
READY.
■
```

```
10 LET C=7
20 PRINT "C"
30 PRINT C
40 END
RUN
C
7
READY.
■
```

Aunque te pueda parecer sorprendente, existe una "operación" llamada *suma de cadenas*, que consiste en poner una detrás de otra. En esta operación se suele utilizar el signo más (+), pero algunos ordenadores utilizan otros signos. Mira el siguiente programa y su resultado en pantalla.

```
10 PRINT "AMAR"+"RAS"
20 PRINT "PAN"+"TALON"
30 END
RUN
AMARRAS
PANTALON
READY.
■
```

3.2 VARIABLES DE CADENA

De la misma forma que los números se pueden guardar en cajitas que se llaman variables, también las cadenas o palabras pueden guardarse en variables. Puedes comprender que son variables distintas: unas admiten operaciones aritméticas y las otras sólo esa suma especial. Para distinguir las *variables de cadena* ha de ponerse detrás de su nombre el signo del dólar ($).

Las variables de cadena también se asignan con LET y se imprimen con PRINT. Pasa por tu ordenador los ejemplos que vienen a continuación:

```
10 LET A$="CAL"
20 LET B$="VINO"
30 PRINT A$+B$
40 END
RUN
CALVINO
READY.
■
```

```
10 LET A$="DAME"
20 LET B$="DINERO"
30 PRINT A$+" "+B$
40 END
RUN
DAME DINERO
READY.
■
```

En el programa de la derecha aparece una cosa nueva. La suma pega dos palabras, pero a veces necesitamos separarlas por un espacio. En la línea 30 hemos incluido un espacio entre comillas para que las palabras no se junten.

3.3 TRABALENGUAS

Prueba a repetir deprisa varias veces la frase:

Pablito clavó un clavito
¿Qué clavito clavó Pablito?

Hablando es fácil equivocarse, ¿verdad? El ordenador, si lo programas bien, no se equivoca nunca. La solución más cómoda sería:

```
10 PRINT "PABLITO CLAVO UN CLAVITO"
20 PRINT "QUE CLAVITO CLAVO PABLITO?"
30 END
```

Vamos a hacer el mismo programa de una forma un poco más complicada, pero que nos va a servir para entender mejor las variables de cadena. Vamos a guardar en variables de cadena las tres palabras que se repiten en las dos frases. Luego, cuando las necesitemos, sacaremos las copias que hagan falta.

```
10 LET P$=" PABLITO"
20 LET C$=" CLAVO"
30 LET T$=" CLAVITO"
40 PRINT P$+C$+" UN"+T$
50 PRINT "¿QUE"+T$+C$+P$+"?"
60 END
```

En las tres primeras líneas se definen las tres palabras que se repiten, y se guardan en variables de cadena. En las líneas 40 y 50 se sacan copias a medida que se necesitan. Ha habido que sumar cadenas, y para que saliera bien, sin juntar palabras, hemos tenido que distribuir bien los espacios en blanco.

Ahora teclea en tu ordenador la línea:

```
45 PRINT
```

Al rodar el programa, verás que las dos frases aparecen separadas por una línea en blanco, ganando en estética. Este es el efecto de la línea 45: la instrucción PRINT sin nada detrás "imprime" una línea en blanco.

Observa que, igual que en PRINT, en LET las cadenas han de ponerse entre comillas.

3.4 INPUT A$

Al intercambiar las dos sílabas de una palabra, a veces resulta otra palabra que significa algo. Por ejemplo:

DOBLAN .... BLANDO
TELA .... LATE
JAMON .... MONJA

Hagamos un programa que al darle las dos sílabas, escriba las dos palabras. Con lo que sabemos ahora podemos poner:

```
10 LET A$="DO"
20 LET B$="BLAN"
```

```
30 PRINT A$+B$
40 PRINT
50 PRINT B$+A$
60 END
```

Al ejecutar este programa sale en la pantalla:

```
DOBLAN

BLANDO
READY.
■
```

Para rodar el programa con las palabras "late" y "jamón", hay que modificar las líneas 10 y 20, pero podemos utilizar la instrucción INPUT, de manera que sea el mismo programa quien pregunte las sílabas, sin necesidad de modificar las líneas. No tiene ninguna dificultad:

```
10 INPUT A$
20 INPUT B$
30 PRINT A$+B$
40 PRINT
50 PRINT B$+A$
60 END
```

Al llegar a la línea 10, el ordenador saca una interrogación. Vamos a contestar SA [RETURN] . Después sale otra interrogación, correspondiente a la línea 20 y escribimos CA [RETURN] . En la pantalla resulta:

```
? SA
? CA
SACA

CASA
READY.
■
```

EJERCICIOS

3.1 Seguro que conoces estos refranes:

```
A1$  QUIEN A BUEN ARBOL SE ARRIMA
A2$  BUENA SOMBRA LE COBIJA
B1$  DIME CON QUIEN ANDAS
B2$  Y TE DIRE QUIEN ERES
C1$  CRIA CUERVOS
C2$  Y TE SACARAN LOS OJOS
```

Haz un programa que asigne a las variables de cadena la frase correspondiente, y que luego escriba los refranes completos. También puedes mezclar una parte de un refrán con otra de otro y construir unos refranes muy originales.

3.2 Haz un programa que, mediante INPUT, guarde en la variable N$ el nombre de una persona; que asigne a F$ la cadena "TE QUIERO" y que, por último, escriba N$ + F$.

3.3 Conocerás el trabalenguas que dice:

El cura de Alcañiz,
a las narices llama nariz
y el cura de Alcañices
a la nariz llama narices

Fíjate que los trozos subrayados se repiten. Puedes asignar cada uno de ellos a una variable de cadena y hacer un programa que escriba el trabalenguas como suma de variables.

3.4 Con las mismas ideas que en el ejercicio anterior, puedes programar este trabalenguas:

La ciudad de Venecia está engondolada,
quién la desengondolará;
el desengondolador que la desengondole
buen desengondolador será.

4

CALCULOS ARITMETICOS

4.1 FORMULAS

El consumo de gasolina super de un coche es 7,8 litros cada 100 kilómetros. Cada litro de gasolina cuesta 93 ptas. Con ayuda del ordenador vamos a realizar los cálculos para saber lo que hemos gastado, si recorremos 415 kilómetros.

Gasolina que se gasta en recorrer un kilómetro:

Fórmula matemática	Fórmula en BASIC
$\frac{7,8}{100}$	7.8/100

Gasolina que se gasta en recorrer 415 kilómetros:

Fórmula matemática	Fórmula en BASIC
$\frac{7,8}{100} \times 415$	7.8/100*415

Dinero gastado:

Fórmula matemática	Fórmula en BASIC
$\frac{7,8}{100} \times 415 \times 93$	7.8/100*415*93

Si queremos obtener el resultado, escribiremos este pequeño programa:

```
10 PRINT 7.8/100*415*93
20 END
```

Escribe RUN y pulsa la tecla [RETURN]. En la pantalla aparecerá:

```
RUN
3010.41
READY.
■
```

4.2 JERARQUIA DE LAS OPERACIONES ARITMETICAS

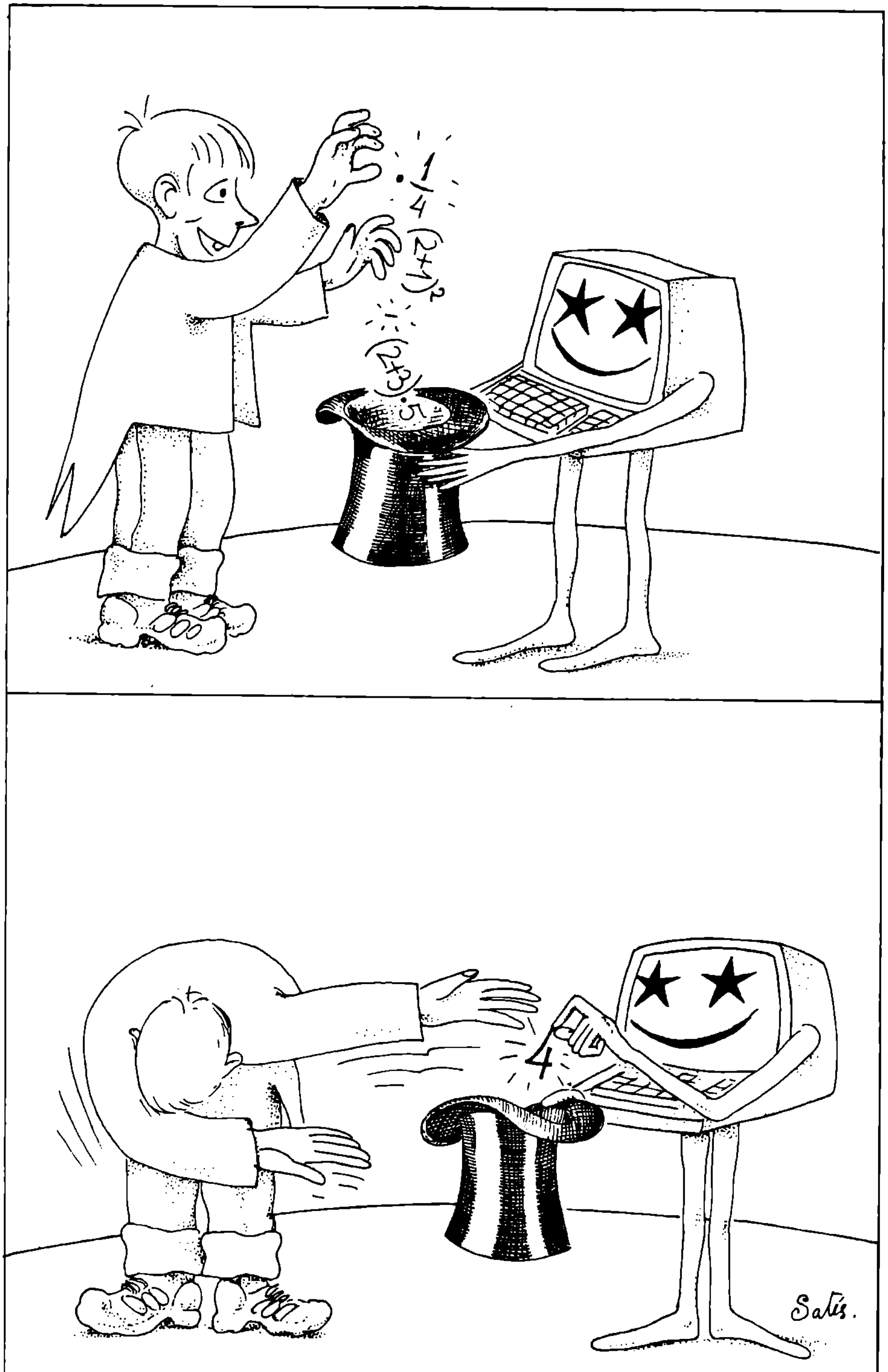
Las operaciones aritméticas son:

Suma (+)
Resta (—)
Multiplicación (*)
División (/)
Elevar a potencias (↑)

La jerarquía de las operaciones aritméticas es:

Primer nivel: ↑
Segundo nivel: * y /
Tercer nivel: + y —

En una fórmula, primero se ejecutan las operaciones del primer nivel, esto es, todas las potencias; después se ejecutan las del segundo nivel (multiplicaciones y divisiones) y, por último, se efectúan las sumas y restas. Dentro de cada nivel las operaciones se realizan de izquierda a derecha.


4
Satis.

Vamos a verlo con varios ejemplos:

Ejemplo 1

	(2) (1)	
	3 + 2 ↑ 3	
(1)	3 + 8	Se ha realizado la potencia en primer lugar.
(2)	11	Se ha realizado la suma en segundo lugar.

Ejemplo 2

		(2)		(3)		(1)		
	3	*	2	+	2	↑	2	
(1)	3	*	2	+		4		1º la potencia.
(2)		6		+		4		2º la multiplicación.
(3)				10				3º la suma.

Ejemplo 3

		(2)		(4)		(1)		(5)		(3)		
	5	*	6	+	3	↑	2	—	14	/	2	
(1)	5	*	6	+		9		—	14	/	2	1º la potencia.
(2)		30		+		9		—	14	/	2	2º la multiplicación.
(3)		30		+		9		—		7		3º la división.
(4)				39				—		7		4º la suma.
(5)								32				5º la resta.

4.3 USO DE PARENTESIS

En el párrafo anterior has visto que las operaciones se realizan según una determinada jerarquía, y entre aquellas operaciones que tienen el mismo nivel se ejecutan antes las que están situadas más a la izquierda.

Para que las operaciones se ejecuten con arreglo a las fórmulas que tenemos, vamos a ver cómo debemos utilizar los paréntesis para alterar ese orden. Veamos un ejemplo:

$$\frac{12}{3} + 2$$

Escribamos un pequeño programa para calcularlo:

```
10 PRINT 12/3+2
20 END
```

El resultado en la pantalla es:

```
RUN
6
READY.
■
```

Si el ejemplo fuese el siguiente:

$$\frac{12}{3+2}$$

la primera operación a realizar sería la suma, y para que el ordenador lo efectúe así, es necesario utilizar los paréntesis. El programa será:

```
10 PRINT 12/(3+2)
20 END
```

Al ejecutarlo, aparece en la pantalla:

```
RUN
2.4
READY.
■
```

Se pueden meter unos paréntesis en otros; el ordenador comenzará la ejecución por los más interiores.

Tomemos la siguiente fórmula:

$$\frac{(2+3)\times 5 - (2+1)^2}{4}$$

Escribámosla en BASIC, y sigamos la ejecución:

		(1)		(4)		(5)		(2)		(3)		(6)		
	((2	+	3)	*	5	—	(2	+	1)	↑	2)	/	4	
(1)	(	5		*	5	—	(2	+	1)	↑	2)	/	4	1º la primera suma.
(2)	(	5		*	5	—		3		↑	2)	/	4	2º la segunda suma
(3)	(	5		*	5	—				9	)	/	4	3º la potencia.
(4)	(			25		—				9	)	/	4	4º la multiplicación.
(5)						16						/	4	5º la resta.
(6)												4		6º la división.

4.4 NOTACION CIENTIFICA

Teclea el siguiente programa:

```
10 PRINT 1000000*1000000
20 END
```

Aparecerá en la pantalla:

```
RUN
1E+12
READY.
■
```

La expresión 1E+12 es una forma abreviada de escribir 1000000000000, que recuerda la notación científica de este número:

$$1 \times 10^{12}$$

Ahora escribe el programa siguiente:

```
10 PRINT 1/100000000
20 END
```

La pantalla presentará:

```
RUN
1E-08
READY.
■
```

En este caso la expresión 1E — 08 es una forma abreviada de escribir 0'00000001, que representa la notación científica del número:

$$1 \times 10^{-8}$$

EJERCICIOS

4.1 Prepara un programa para que el ordenador pida (INPUT) la base B y la altura A de triángulo y que luego escriba (PRINT) el área S.

$$S = \frac{1}{2} \; B \times A$$

4.2 Un corredor hace los 100 metros en 9,8 segundos. Haz un programa para obtener la velocidad en kilómetros/hora

$$V=\frac{100\times 3600}{9,8\times 1000}$$

4.3 Haz dos programas que escriban el valor decimal de la fracción

$$F=\frac{\dfrac{3}{5}+\dfrac{7}{4}}{\dfrac{9}{8}+\dfrac{5}{2}}$$

En el primer programa llamas N al numerador y D al denominador, y después calculas F = N/D.

En el segundo puedes hallar F directamente, sin variables intermedias (¡cuidado con los paréntesis!).

Comprueba que en ambos casos obtienes el mismo resultado.

4.4 Imagina un micromercado en el que únicamente se vende arroz, garbanzos y lentejas, al precio de 117 ptas., 129 ptas. y 133 ptas. el kilo, respectivamente. La cajera tiene un microordenador, pero no tiene un programa que haga la cuenta a cada cliente. ¿Por qué no le haces tú el programa?

5

TRES FUNCIONES IMPORTANTES

5.1 LAS FUNCIONES SQR, INT Y ABS

Además de las operaciones aritméticas +, —, *, etc., los ordenadores, lo mismo que las calculadoras de bolsillo, pueden hacer otras operaciones, como, por ejemplo, extraer la raíz cuadrada, hallar la parte entera o calcular el valor absoluto.

5.2 LA RAIZ CUADRADA

El lenguaje BASIC, para hallar raíces cuadradas, posee la función SQR (SQuare Root = raíz cuadrada). Por ejemplo, si quieres calcular $\sqrt{4}$, tienes que poner SQR(4). Teclea el siguiente programa:

```
10 LET A=25
20 PRINT "NUMERO"
30 PRINT A
40 PRINT "RAIZ CUADRADA"
50 PRINT SQR(A)
60 END
```

En la pantalla aparecerá:

```
NUMERO
25
RAIZ CUADRADA
5
READY.
■
```

El número que pones entre paréntesis siempre debe ser positivo o cero, pero nunca negativo. Si el número es negativo, el ordenador te sacará un mensaje de error. Vamos a probarlo.

Si en la línea 10 pones lo siguiente:

```
10 LET A=-4
```

en la pantalla saldrá:

```
NUMERO
-4
? ILLEGAL QUANTITY
ERROR IN 50
READY.
■
```

5.3 LA PARTE ENTERA

En principio, esta función te parecerá bastante tonta, sin embargo irás descubriendo su utilidad a lo largo del libro. Lo que hace es sencillo:

```
INT(4.68547)=4   INT(3.1415926)=3   INT(6)=6
```

La parte entera quita los decimales de un número. Para que lo comprendas mejor, teclea el programa siguiente:

```
10 LET A=8.762
20 PRINT "NUMERO"
30 PRINT A
40 PRINT "PARTE ENTERA"
50 PRINT INT(A)
60 END
```

En la pantalla nos aparecerá:

```
NUMERO
8.762
PARTE ENTERA
8
READY.
■
```

5.4 EL VALOR ABSOLUTO

En una ciudad el termómetro marca 15 grados y en otra ciudad 17. Usualmente, decimos que hay una diferencia de 2 grados porque prescindimos del signo menos (—). Esto también lo puede hacer el ordenador con la función ABS (valor absoluto). Así, por ejemplo:

```
ABS(-4.37)=4.37   ABS(-8)=8   ABS(3)=3
```

Escribamos el siguiente programa:

```
10 LET A=15
20 PRINT "TEMPERATURA PRIMERA CIUDAD"
30 PRINT A
40 LET B=17
50 PRINT "TEMPERATURA SEGUNDA CIUDAD"
60 PRINT B
70 LET C=A-B
80 PRINT "DIFERENCIA TEMPERATURAS"
90 PRINT ABS(C)
100 END
```

En la pantalla tendremos:

```
TEMPERATURA PRIMERA CIUDAD
15
TEMPERATURA SEGUNDA CIUDAD
17
DIFERENCIA TEMPERATURAS
2
READY.
■
```

EJERCICIOS

5.1 Del famoso teorema de Pitágoras se deduce que "la hipotenusa es igual a la raíz cuadrada de la suma de los cuadrados de los catetos". Haz un programa que calcule la hipotenusa de triángulo rectángulo, tomando como datos los valores de los dos catetos.

5.2 Calixto es un programador listo. Ayuda a las cooperativas de productores a repartir los beneficios entre todos los socios. No cobra nada por el programa, se conforma con los céntimos de peseta que sobran en los repartos; pues, como dice, la peseta es la moneda más pequeña y menos de eso no se puede pagar.

Haz un programa para repartir un millón de pesetas de beneficios entre los 1.997 socios de una cooperativa, que diga cuánto le corresponde a cada socio y cuánto a Calixto el listo.

5.3 Tu ordenador no sabe hallar el cociente y el resto de una división entera de dos números (de 370 dividido por 13, por ejemplo). Explícale cómo puede hacer esa división.

5.4 El resultado de una operación puede aparecer en la pantalla con muchas cifras decimales, pero en la mayor parte de los casos nos interesa que aparezcan únicamente 2 decimales. Supongamos, como ejemplo, que el número N vale 9.12345678 y que nosotros deseamos imprimir el número 9.12 ¿Cómo quedarnos con estas cifras y prescindir de las otras cifras decimales de N? Haz estas operaciones:

1) Multiplica N por 100: 100×N=912.345678
2) Aplica la función INT: INT(912.345678)=912
3) Divide por 100: 912/100=9.12

Ya tienes un número con dos decimales. Pasar del número N = 9.12345678 al número T = 9.12, se llama *truncar* el número N a dos cifras decimales.

Fíjate bien en las tres operaciones que has realizado y escribe una instrucción que trunque cualquier número N a dos cifras decimales.

6

MANEJO DE DATOS

6.1 COMO UNA CALCULADORA

Para pequeños cálculos no es necesario que hagas un programa; puedes utilizar el ordenador como si fuese una calculadora, escribiendo las instrucciones sin número de línea. Entonces el ordenador las ejecuta inmediatamente, sin esperar al RUN, con sólo pulsar [RETURN] . A esta manera de trabajar se le llama modo inmediato o comando. Prueba los siguientes ejemplos:

```
PRINT 7*3-5       [RETURN]
16
READY.
PRINT 8↑2/3        [RETURN]
21.3333333
READY.
■
```

De esta forma el ordenador funciona igual que una calculadora, siempre que pongamos delante la instrucción PRINT. Con la instrucción LET puede hacer lo mismo, y se comporta entonces como las memorias de las calculadoras:

```
LET M=57*8   [RETURN]
READY.
PRINT M   [RETURN]
456
READY.
PRINT (M-3)/3   [RETURN]
151
READY.
■
```

M = 456

(456 — 3)/3 = 453/3 = 151

No insistiremos sobre esto, pero la mayoría de las instrucciones del BASIC, (la instrucción INPUT es una excepción), pueden ser tratadas en modo inmediato o comando.

6.2 READ-DATA

La forma de introducir datos en el ordenador depende de cada situación. Si es un dato que queremos modificar fácilmente, lo mejor será un INPUT. Si va a ser más o menos fijo, será un LET. Para un número grande de datos es preferible utilizar las instrucciones READ y DATA que veremos ahora.

```
10 DATA SEAT-RONDA,929504,PEUGEOT-205,860815
20 DATA OPEL-CORSA,722349,RENAULT-11,1018230
30 DATA VOLKSWAGEN-POLO,834722
40 READ A$,A
50 READ B$,B
60 READ C$,C,D$
70 READ D,E$,E
80 PRINT A$,A
90 PRINT B$,B
100 PRINT C$,C
110 PRINT D$,D
120 PRINT E$,E
130 PRINT (A+B+C+D+E)/5
140 END
```

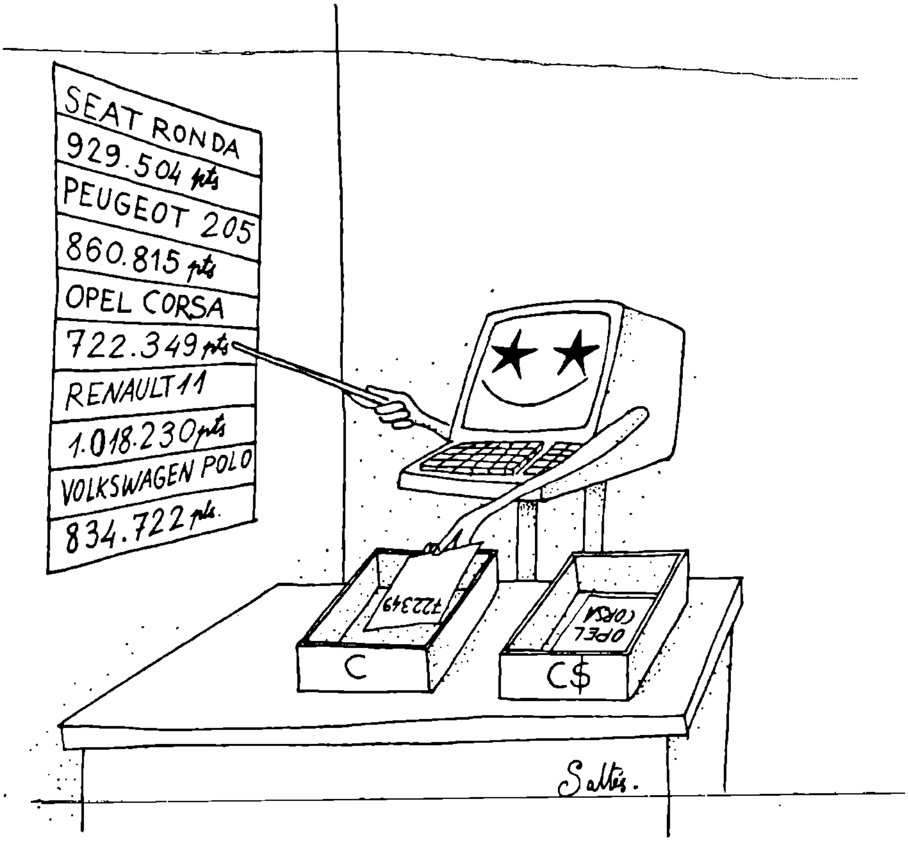
Este programa saca en la pantalla cinco modelos de coches y sus precios. Al final imprime el precio medio. Las comas (,) utilizadas en las instrucciones PRINT separan los valores que se presentan en la pantalla.

Las líneas DATA forman una especie de pila o almacén de datos ordenados, separados por comas (,). No hay que preocuparse de nada más. No importa que la línea 20 tenga cuatro datos y la 30 dos; lo que importa es que el dato número 3 está detrás del número 2 y antes del número 4. Aunque los datos 860815 y OPEL-CORSA están en líneas distintas, el ordenador los trata como si estuvieran separados por una coma (,). Daría igual poner los datos de coches en cuatro líneas en vez de en tres, siempre que el orden sea el mismo. Da lo mismo poner:

```
10 DATA SEAT-RONDA,929504,PEUGEOT-205,860815
20 DATA OPEL-CORSA,722349,RENAULT-11,1018230
30 DATA VOLKSWAGEN-POLO,834722
```

que poner:

```
10 DATA SEAT-RONDA,929504,PEUGEOT-205
20 DATA 860815,OPEL-CORSA,722349
30 DATA RENAULT-11,1018230
35 DATA VOLKSWAGEN-POLO,834722
```



La instrucción READ de la línea 40 busca el primer dato y lo guarda en A$; y el siguiente dato lo guarda en A. En la línea 50 se busca el siguiente dato y se guarda en B$; y el siguiente se guarda en B.

Como ves, READ lee el siguiente dato, o sea, el primer dato no leído. Robotín lleva la cuenta de los datos que se leen; tiene un puntero que señala el próximo dato a leer, y cada vez que se lee algo, baja el puntero. Al principio el puntero está en SEAT-RONDA.

En la instrucción READ se pueden poner varias variables separadas por comas (,) igual que en DATA. En las líneas 40 y 50 hemos puesto dos, y en las siguientes, tres. READ tiene la misma flexibilidad que DATA. Da lo mismo poner:

```
30 READ A$,A
40 READ B$,B
```

que poner:

```
30 READ A$,A,B$
40 READ B
```

Los precios de los coches suelen variar cada año. Cada vez que cambien habrá que modificar las líneas DATA. No es mucho trabajo tener estos datos actualizados. Para este tipo de información son adecuadas las instrucciones READ-DATA.

6.3 POSIBLES ERRORES Y OTROS ASPECTOS

No es necesario leer todo el bloque de datos entero; se puede dejar a medio leer. Por el contrario, si hay 10 datos y pretendemos leer más, el ordenador sacará un mensaje de error como:

OUT OF DATA ERROR

Otro posible error es tratar de leer con una variable numérica una cadena. Si le llega el turno, por ejemplo, a la instrucción READ B, y el puntero está en OPEL-CORSA, sale un mensaje de error porque OPEL-CORSA se puede meter en B$, pero no en B.

Las líneas DATA no se ejecutan; su función se limita a formar el almacén ordenado de datos. Es recomendable ponerlas juntas, en bloque, al principio o al final del programa.

Dentro de un programa la pila de DATA puede leerse varias veces si queremos. Para eso es necesario volver a colocar el puntero al principio, en SEAT-RONDA. Con la instrucción RESTORE, Robotín pone el puntero en el primer dato, y permite así recomenzar la lectura.

EJERCICIOS

6.1 En un programa tenemos las siquientes líneas:

```
100 DATA DUERO,TAJO,GUADALQUIVIR
110 DATA EBRO,SEVILLA,ZAMORA
120 DATA ZARAGOZA,TOLEDO
```

Haz un programa que nos escriba en líneas sucesivas el río y la ciudad por la que pasa.

6.2 Prepara un programa que lea (READ) los nombres de tres compañeros (DATA); que también lea sus notas de matemáticas; y que, por último, escriba los nombres junto con la nota correspondiente.

6.3 Guarda en líneas DATA los nombres y fechas de nacimiento de los miembros de tu familia. Completa el programa para que lea los datos y los escriba ordenadamente.

6.4 En dos líneas DATA pon el nombre y la dirección de dos amigos. Diseña un programa para que lea los datos y los escriba en forma de dirección postal (como si la pantalla fuera un sobre).

7

CLARIDAD DE EXPRESION

7.1 COMENTARIOS Y OBSERVACIONES

Cuando hayas coleccionado un buen paquete de programas y pase el tiempo, te puede costar recordar para qué sirve cada uno y cómo funciona. Quizá hagas anotaciones a mano en los listados para explicarlos. Pues bien, esas aclaraciones se pueden meter como parte del programa mediante la instrucción REM, que es abreviatura de Remark (comentario, observación).

En adelante conviene que pongas título a los programas, empezándolos por una instrucción REM, como por ejemplo:

```
10 REM TABACO
20 REM GASOLINA
30 REM TRABALENGUAS
```

La instrucción REM no se ejecuta. Sólo sirve para figurar en el listado y explicar el programa. Cuando al rodar el programa el ordenador encuentra un REM, no sigue leyendo esa línea y pasa de largo. Detrás de REM puedes escribir lo que quieras, incluso felicitaciones o insultos a Robotín.

Los programas que publican los libros o revistas suelen tener varias líneas REM, explicando las distintas partes, o los trucos utilizados. Cuando tengas que hacer un programa que vayan a leer los demás procura explicar los pasos principales con instrucciones REM.

7.2 EL PUNTO Y COMA

Hasta ahora hemos visto tres posibilidades de la instrucción PRINT:

`PRINT "HOLA" o PRINT A$`	imprime una cadena
`PRINT 7 o PRINT 3*X/5`	imprime un número
`PRINT`	"imprime" una línea en blanco

Una sola instrucción PRINT puede imprimir también varias cosas si se separan por puntos y comas (;). Por ejemplo:

```
10 PRINT "A";"B";"C"
20 END
```

imprime:

```
ABC
READY.
■
```

En BASIC la pantalla tiene una especie de cuadrícula invisible. El ordenador no escribe como le da la gana, sino que se ajusta a la cuadrícula, poniendo en cada cuadradito un carácter (letra o número) o un espacio en blanco. Mira el siguiente programa y su resultado en pantalla:

```
10 PRINT "A";"B";"C"
20 PRINT "D";"E"
30 PRINT "F";"G";"H";"I"
40 END
```

```
ABC
DE
FGHI
READY.
■
```

El punto y coma (;) en PRINT ordena "imprimir a continuación". Si escribimos en una instrucción PRINT varias cosas, una detrás de otra, y separadas por puntos y comas (;), al ejecutar el programa se imprimen todas seguidas.

El punto y coma (;) puede ponerse también al final de la línea PRINT, y entonces ordena que la siguiente instrucción PRINT se escriba junto a ella. Lo entenderás mejor con un ejemplo:

```
10 PRINT "A";"B";"C";        (punto y coma final)
20 PRINT "D";"E"
30 PRINT "F";                (punto y coma final)
40 PRINT "G"
50 PRINT "H"
60 PRINT "I";"J";"K"
70 END
```

```
ABCDE
FG
H
IJK
READY.
■
```

7.3 MARCADOR ELECTRONICO

Los buenos estadios de fútbol tienen un marcador electrónico. Vamos a tratar de manejarlo en BASIC.

```
10 REM MARCADOR ELECTRONICO
20 PRINT "REAL MADRID";3;"VALENCIA"; 1
30 PRINT "ATL. MADRID";1;"BARCELONA"; 2
40 PRINT "REAL SOCIEDAD";1;"SEVILLA"; 0
:  :     :       :   :      :
```

Este programa imprime:

R	E	A	L		M	A	D	R	I	D		3		V	A	L	E	N	C	I	A		1								
A	T	L	.		M	A	D	R	I	D		1		B	A	R	C	E	L	O	N	A		2							
R	E	A	L		S	O	C	I	E	D	A	D		1		S	E	V	I	L	L	A		0							

El PRINT de la línea 20 escribe las cuatro cosas que están separadas por puntos y comas (;). Primero escribe REAL MADRID, a continuación un 3, después VALENCIA y para terminar un 1. Las cadenas (los equipos) van entre comillas, y los goles (números) se ponen directamente.



Conviene que sepas por qué han salido unos espacios en blanco junto a los números, es decir, por qué ha salido:

```
R E A L   M A D R I D ↓ 3 ↓ V A L E N C I A ↓ 1
```

en vez de:

```
R E A L  M A D R I D 3 V A L E N C I A 1
```

La razón es que los números llevan signo. Si el signo es menos (—), se pone, y si es más (+), se deja un espacio en blanco. Piensa entonces que delante del 3 y del 1 hay un signo más (+), que es sustituido por un espacio en blanco. Si por un absurdo los goles fueran negativos, entonces habría salido:

```
R E A L   M A D R I D - 3 ↓ V A L E N C I A - 1
```

Queda explicar el espacio entre 3 y VALENCIA. Para facilitar la claridad, muchas máquinas dejan siempre detrás de los números un espacio en blanco. Otras no lo hacen, y para que no salga:

```
R E A L  M A D R I D  3 V A L E N C I A  1
```

tendrás que añadir tú el espacio en blanco, por ejemplo así:

```
20 PRINT "REAL MADRID";3;" VALENCIA";1
```

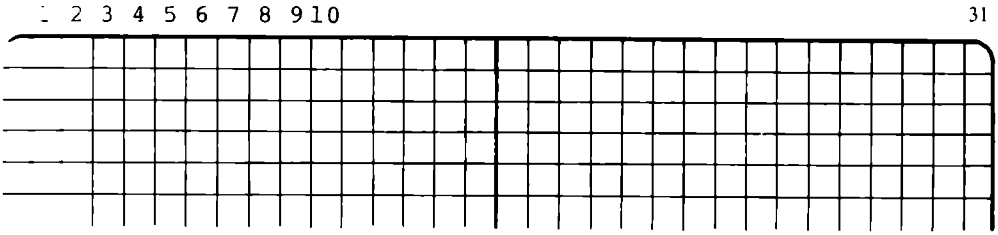
o así:

```
20 PRINT "REAL MADRID";3;" ";"VALENCIA";1
```

Como ves, el punto y coma (;) se parece mucho a la suma de cadenas que ya conoces.

7.4 LA FUNCION PRINT TAB

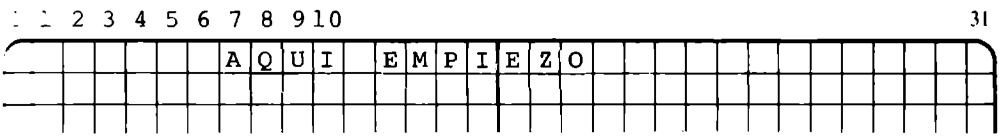
En la cuadrícula invisible que antes te contaba, las columnas están numeradas.



Aunque te extrañe, normalmente los ordenadores empiezan a numerar por el cero, de modo que la primera columna es la 0, la segunda la 1 y así hasta el final. La instrucción:

```
60 PRINT TAB(7);"AQUI EMPIEZO"
```

Imprime "AQUI EMPIEZO" a partir de la columna 7.

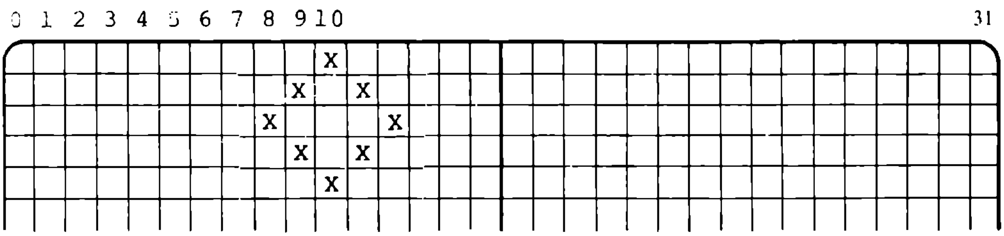


Esto es lo que hace la instrucción PRINT TAB: escribir a partir de una columna. El número de la columna se pone detrás de TAB entre paréntesis, y a continuación se pone punto y coma (;).

El programa:

```
10 REM TAB
20 PRINT TAB(10);"X"
30 PRINT TAB(9);"X X"
40 PRINT TAB(8);"X   X"
50 PRINT TAB(9);"X X"
60 PRINT TAB(10);"X"
70 END
```

saca en la pantalla:



Puedes escribir con PRINT TAB todos los ejemplos de programas que has visto desde el principio del libro. Así, en el programa de los precios de los coches del apartado 6.2 puedes modificar las líneas PRINT para que queden así:

```
80 PRINT A$;TAB(17);A
90 PRINT B$;TAB(17);B
100 PRINT C$;TAB(17);C
110 PRINT D$;TAB(17);D
120 PRINT E$;TAB(17);E
```

y al ejecutar el programa, aparecerá en la pantalla:

```
SEAT-RONDA       929504
PEUGEOT-205      860815
OPEL-CORSA       722349
RENAULT-11       1018230
VOLKSWAGEN-POLO  834722
```

7.5 BUENA PRESENTACION

En los marcadores electrónicos los equipos y los resultados salen alineados, en orden. Por ejemplo así:

```
0 1 2 3 4 5 6 7 8 910                                    31
REAL MADRID   3 VALENCIA  1
ATL. MADRID   1 BARCELONA 2
REAL SOCIEDAD 1 SEVILLA   0
```

Es fácil de conseguir. ¿Has manejado alguna vez el tabulador de una máquina de escribir? La instrucción PRINT TAB hace lo mismo que el tabulador. La buena presentación anterior se consigue con un programa como el siguiente:

```
10 REM MARCADOR ELECTRONICO
20 PRINT "REAL MADRID";TAB(13);3; "VALENCIA";TAB
   (25);1
30 PRINT "ATL. MADRID";TAB(13);1; "BARCELONA";TA
   B(25);2
40 PRINT "REAL SOCIEDAD";TAB(13);1;"SEVILLA";TAB
   (25);0
```

La función TAB hace que los goles del Real Madrid no salgan pegados a él, sino en la columna 13, como señala TAB (en 13 sale el signo más (+) que no se pone, y en el 14 el número 3). De igual manera el 1 no sale junto a VALENCIA, sino en la columna 25 (en 25 sale el signo más (+), que no se pone, y en 26 el número 1). Poniendo en todas las líneas TAB(13) y TAB(25) salen todos los resultados alineados.

Hemos trabajado con una pantalla de 32 columnas. Cuenta el número de columnas de la pantalla de tu ordenador para presentar bien los resultados: que quepan, que queden centrados, etc.

7.6 LA COMA

En PRINT también pueden utilizarse comas (,) de forma parecida a los puntos y comas (;). La coma (,) separa más en la pantalla los distintos letreros o números a escribir. Pero la coma (,) tiene reglas algo más complicadas, varía algo según el modelo de microordenadores y además resulta poco práctica, apenas se usa. Si quieres informarte bien del funcionamiento de las comas (,), puedes encontrarlo en el libro BASIC BASICO *Curso de programación.* Capítulo 3, página 56 y siguientes.

7.7 LOS DOS PUNTOS

En una línea se pueden poner varias instrucciones separadas por dos puntos (:). Por ejemplo:

```
10 PRINT "HOLA"
20 PRINT:PRINT:PRINT
30 PRINT "ADIOS"
40 END
```

En la pantalla sale:

```
HOLA


ADIOS
READY.
■
```

Esto vale para todas las instrucciones, no sólo para PRINT. Mira otro ejemplo:

```
10 LET A=5:LET B=7:LET C=12
20 PRINT A;"+";B;"=";12
30 PRINT:PRINT
40 LET A=B*C-10:LET C=B↑2-1
50 PRINT A;B;C
60 END
```

Produce el siguiente resultado en la pantalla.

```
5+7=12


74 7 48
```

No debes confundir el papel de los dos puntos (:) con el del punto y coma (;) o la coma (,).

— Los dos puntos (:) permiten varias instrucciones en una línea, y precisamente marcan su separación. Por ejemplo:

```
10 PRINT 1:PRINT 2:PRINT 3
```

son varias instrucciones.

— El punto y coma (;) separa los contenidos de una instrucción PRINT:

```
10 PRINT 1;2;3
```

En el próximo capítulo verás otros usos del punto y coma (;) y de la coma (,), pero también dentro de una sola instrucción.

¿Cuándo debes utilizar los dos puntos y cuándo no? En general, utilízalos poco. Es cuestión de claridad. Una regla orientativa puede ser el agrupar en una línea varias instrucciones del mismo tipo y sencillas (varios LET por ejemplo). No conviene mezclar instrucciones distintas (LET, INPUT, PRINT) en una línea, porque se pierde claridad. La práctica te irá aconsejando el uso más conveniente de los dos puntos (:).

EJERCICIOS

7.1 Debes acostumbrar a tu ordenador a que sea educado con tus amigos. Prográmalo para que pregunte:

```
COMO TE LLAMAS?
```

y que, inmediatamente de recibir el nombre de tu amigo (LUIS), conteste:

```
ME ALEGRO DE CONOCERTE LUIS
```

7.2 Utiliza instrucciones PRINT con coma (,) para que tu ordenador escriba:

```
FRANCIA       PARIS
ITALIA        ROMA
PORTUGAL      LISBOA
```

7.3 Haz uso de la función TAB para que el pez del ejercicio 1.1 quede centrado en la pantalla.

7.4 Prepara un programa que pida el nombre y la dirección postal de una persona (calle, número, código postal y población). Y que luego lo escriba (TAB) en forma de sobre:

```
MANUEL DIAZ RUIZ
TRAFALGAR, 123
28010 MADRID
      ESPAÑA
```

8

CLARIDAD EN LOS DATOS

8.1 ETIQUETAS EN LOS INPUT

En el último capítulo has estudiado la instrucción PRINT en toda su potencia. Ya sabes cuándo hacen falta comillas (") y cuándo no; y sabes combinar el punto y coma (;) con la función TAB. También la instrucción INPUT admite poner comillas ("), comas (,), y puntos y comas (;), pero con usos y fines bastante diferentes. Vamos a trabajar con las estaturas y pesos de personas:

```
10 REM ESTATURA Y PESO
20 INPUT "NOMBRE";N$
30 INPUT "ESTATURA";E
40 INPUT "PESO";P
50 PRINT "NOMBRE";TAB(15);"ESTATURA   PESO"
60 PRINT N$;TAB(16);E;TAB(25);P
70 END
```

Etiquetas

En los INPUT, antes de las variables, hemos puesto una etiqueta o letrero entre comillas ("), separado de la variable por un punto y coma (;). Ahora, al ejecutar el programa, antes de la interrogación sale el letrero. Es la forma de saber qué pregunta la interrogación.

```
RUN
NOMBRE? IGNACIO
ESTATURA? 1.78
PESO? 76
NOMBRE                 ESTATURA        PESO
IGNACIO                  1.78           76
READY.
■
```

Es muy recomendable, por claridad, poner siempre etiquetas en los INPUT. En algún ordenador que no las permite puedes usar el siguiente truco:

```
15 PRINT "NOMBRE";
20 INPUT N$
25 PRINT "ESTATURA";
30 INPUT E
35 PRINT "PESO";
40 INPUT P
```

punto y coma final

La presentación en pantalla de las preguntas es la misma, porque el punto y coma (;) final del PRINT obliga a que la interrogación del INPUT salga a continuación del letrero.

8.2 VARIAS VARIABLES EN UN INPUT

Se puede abreviar el programa haciendo las tres preguntas en una sola instrucción INPUT:

```
10 REM ESTATURA Y PESO
20 PRINT "NOMBRE, ESTATURA Y PESO";
30 INPUT N$,E,P
40 PRINT "NOMBRE";TAB(15);"ESTATURA PESO"
50 PRINT N$;TAB(16);E;TAB(25);P
60 END
```

Hemos metido las tres preguntas en la línea 30 en la que se ven las tres variables separadas por comas (,). No importa que unas variables sean numéricas y otras de cadena.

Las comas (,) en el INPUT de la línea 30 del programa, deben corresponderse con comas (,) de separación de las respuestas.

comas

```
30 INPUT N$,E,P
```

```
RUN
NOMBRE, ESTATURA Y PESO? IGNACIO,1.78,76
NOMBRE          ESTATURA   PESO
IGNACIO           1.78      76
```

comas

Y así la primera respuesta (IGNACIO) va a la primera variable (N$). Tras la coma (,) la segunda respuesta (1.78) va a la segunda variable (E). Y por último 76 va a P. Hay que separar las respuestas por comas (,), igual que las variables del INPUT.

Tienes que tener cuidado en escribir el mismo número de datos que los que te piden (en este caso tres), ni más ni menos. Si escribimos:

```
NOMBRE, ESTATURA Y PESO? IGNACIO,1.78 [RETURN]
```

2 respuestas

o también:

```
NOMBRE,ESTATURA Y PESO?IGNACIO,1.78,76,MIGUEL [RETURN]
```

4 respuestas

hay un mensaje de error porque faltan o sobran datos.

Prueba qué diferencia hay entre los dos programas siguientes:

```
10 INPUT A,B
20 PRINT A*B
30 END
```

```
10 INPUT A:INPUT B
20 PRINT A*B
30 END
```

8.3 SINTAXIS GENERAL DEL INPUT

Pueden fundirse en una sola instrucción INPUT la posibilidad de utilizar etiquetas y de manejar varias variables. El programa quedaría:

```
                                          punto y coma
                                          de separación
10 REM ESTATURA Y PESO
20 INPUT "NOMBRE, ESTATURA Y PESO"; N$, E, P
.    .
.    .       etiqueta inicial          variables separadas
.    .       entre comillas            entre sí por comas
```

La línea 20 es la sintaxis (ortografía) más general de la instrucción INPUT. Lleva una etiqueta al principio, un punto y coma (;) de separación y varias variables separadas por comas (,). Fíjate que el letrero va siempre al principio. Es un error de sintaxis poner:

```
20 INPUT "NOMBRE";N$,"ESTATURA";E,"PESO";P
```

Sólo se admite una etiqueta inicial.

EJERCICIOS

8.1 En un programa aparece esta línea:

```
20 INPUT "NOMBRE, CIUDAD, CALLE";N$;CI;CA
```

¿Observas en ella algún error?

8.2 Prepara un programa para que el ordenador calcule el área de un triángulo. Para meter los datos utiliza los INPUT con etiqueta.

8.3 Haz un programa para que el ordenador te pida (INPUT) el NOMBRE, APELLIDOS, NUMERO DE TELEFONO Y ESTADO CIVIL de una persona. Debes colocar etiquetas en las líneas INPUT para que sepas qué te está pidiendo el ordenador. Completa el programa para que escriba esos datos en la pantalla.

8.4 Diseña un programa para que el ordenador te pida la MARCA, el MODELO y la MATRICULA de un coche (todo en una sola línea INPUT) y a continuación que lo escriba en tres líneas.

9

ALGORITMOS

9.1 RECETA PARA HACER LA ENSALADA

Vamos a hacer una ensalada. Necesitamos ingredientes —tomate, lechuga, aceitunas, cebolla, aceite, sal y vinagre— y un cacharro donde ponerlos una vez limpios, partidos y pelados. Para contarlo de una manera más clara, escribimos:

<u>Preparar</u> Ensaladera

<u>Preparar</u> Tomate, Lechuga, Aceitunas, Cebolla, Aceite, Sal y Vinagre.

<u>Lavar</u> el Tomate y la Lechuga

<u>Pelar</u> la Cebolla

<u>Partir</u> el Tomate, la Lechuga y la Cebolla en la Ensaladera

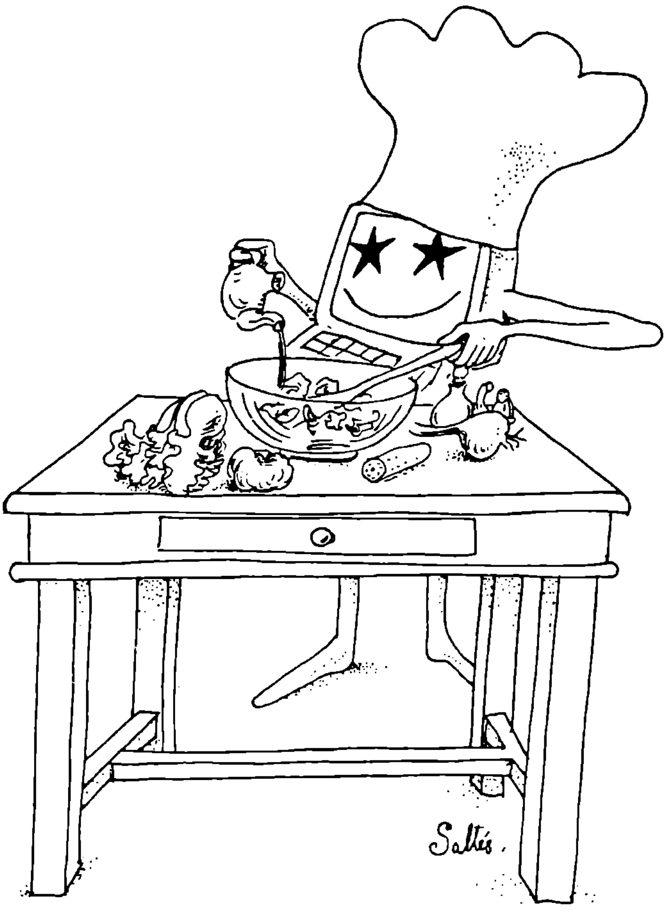
<u>Meter</u> el Tomate, la Lechuga y la Cebolla en la Ensaladera

<u>Poner</u> Aceitunas

<u>Añadir</u> Aceite, Sal y Vinagre

<u>Revolver</u>

<u>Fin</u>



A esta manera de contar las tareas le llamamos *algoritmo.*

Un algoritmo es una forma de describir una acción, tan clara y precisa que cualquiera pueda realizarla correctamente, sin más que seguir con cuidado las instrucciones. Hasta las máquinas ejecutan algoritmos.

Los algoritmos que nos interesan a nosotros son los que pueden realizar los ordenadores. A un ordenador le decimos algo, le damos datos, y él los transforma para proporcionarnos una respuesta.

9.2 ACCIONES ELEMENTALES O MODULOS

Suponte que eres el encargado de la caja de una librería. Para cobrar a un cliente haces lo siguiente: enterarte del precio del libro, decírselo al cliente, recibir el dinero (supongamos que está justo) y registrar el libro como vendido. Este proceso se describe de manera clara y precisa así:

<u>Leer</u> Precio en un catálogo [Precio]

<u>Escribir</u> factura [Precio]

<u>Ingresar</u> el dinero en caja [Dinero]

<u>Registrar</u> el libro como vendido [Libro]

<u>Fin</u>

Este es otro algoritmo. Como ves, hemos descompuesto la acción total en varias partes que llamaremos *módulos*. En cada uno de los módulos hemos subrayado el verbo que indica la acción principal de ese módulo. Si creemos que es necesario, añadimos entre corchetes comentarios o los nombres de las variables que intervienen.

9.3 DIAGRAMAS DE FLUJO

En ocasiones, tan necesaria o más que una descripción literal, puede ser una descripción gráfica, que nos indique visualmente cuál es la estructura del algoritmo que nos interesa. Esta descripción gráfica se llama *diagrama de flujo*.

Volviendo al algoritmo del cajero, el diagrama de flujo será:

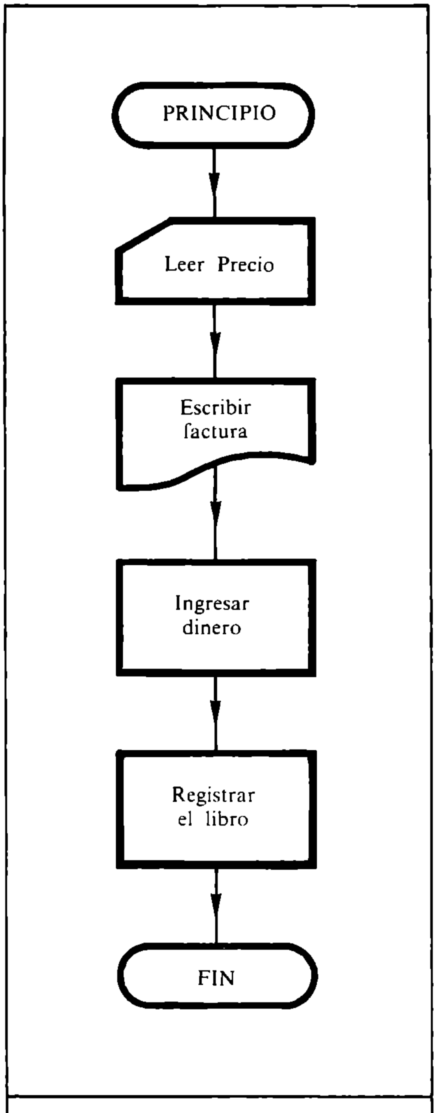


Diagrama de flujo del CAJERO

Las dos elipses simbolizan el principio y el fin. Los rectángulos, operaciones o transformaciones de algún tipo. A las llegadas de información a la persona o máquina que realiza el proceso, se les simboliza por  y a las salidas de información por 

EJERCICIOS

9.1 Escribe en forma de algoritmo lo que haces desde que te levantas hasta que vas al colegio.

9.2 Dado un campo de fútbol escribe en forma de algoritmo las acciones que hay que realizar para obtener su área.

9.3 Freír un huevo requiere una serie de operaciones. Escríbelas en forma de algoritmo.

9.4 Escribe un algoritmo para pegar un sello de correos en un sobre.

10

SALTOS OBLIGATORIOS

10.1 EL CUENTO DE CACARAVACA

Programemos el Cuento de Cacaravaca. Ya sabes cómo es. Preguntas "¿Quieres que te cuente el Cuento de Cacaravaca que nunca se acaba?". Tanto si la contestación es "Sí", como si es "No", tú repites "No quiero que me digas que sí ni que me digas que no. Yo quiero que me digas que si quieres que te cuente el Cuento de Cacaravaca que nunca se acaba" y vuelta a empezar...

El algoritmo del cuento es éste:

1. Decir "¿Quieres que te cuente el Cuento de Cacaravaca que nunca se acaba?"

2. Repetir el recuadro

> Recibir Contestación
>
> Decir "No quiero que me digas que sí ni que me digas que no. Yo quiero que me digas que si quieres que te cuente el Cuento de Cacaravaca que nunca se acaba."

3. Fin



El diagrama de flujo de este cuento es el siguiente:

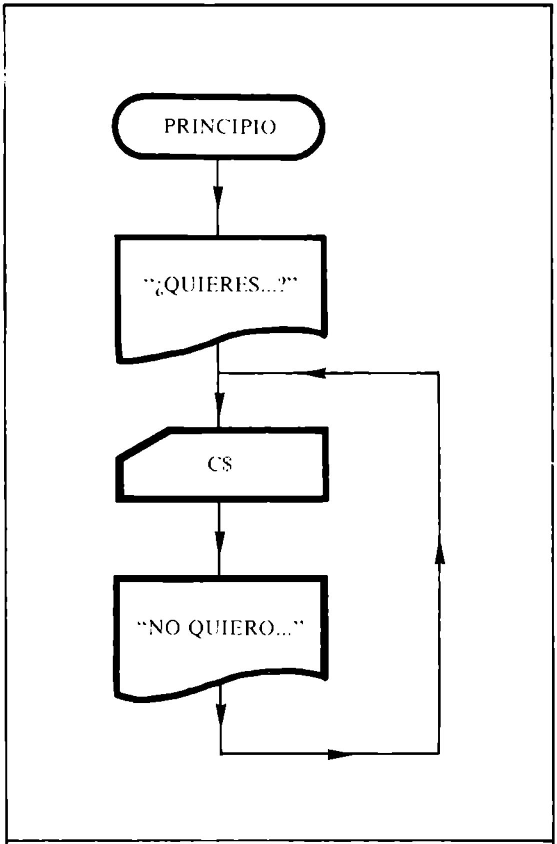


Diagrama de flujo del
CUENTO DE CACARAVACA

10.2 LA INSTRUCCION GOTO

Un programa para el Cuento de Cacaravaca puede ser así:

```
10 REM CUENTO DE CACARAVACA
20 PRINT "QUIERES QUE TE CUENTE EL CUENTO DE CAC
   ARAVACA QUE"
30 PRINT "NUNCA SE ACABA?"
40 INPUT C$
50 PRINT "NO QUIERO QUE ME DIGAS QUE SI NI QUE M
   E DIGAS QUE"
60 PRINT "NO. YO QUIERO QUE ME DIGAS QUE SI QUIE
   RES QUE TE"
70 PRINT "CUENTE EL CUENTO DE CACARAVACA QUE NUN
   CA SE ACABA."
80 GOTO 40
90 END
```

Observa la línea:

```
80 GOTO 40
```

cada vez que el ordenador pasa por ella, se ve obligado a ir a la línea 40 (en inglés, GOTO 40 significa "ir a 40").

Con este programa el proceso no tiene fin, porque el ordenador al llegar a la línea 80 es enviado a la 40, desde donde vuelve a la 80 y de aquí a la 40, y así indefinidamente.

10.3 CONTADOR

Prueba el siguiente programa en tu ordenador:

```
10 REM CONTADOR
20 LET N=0
30 LET N=N+1
40 PRINT N
50 GOTO 30
60 END
```

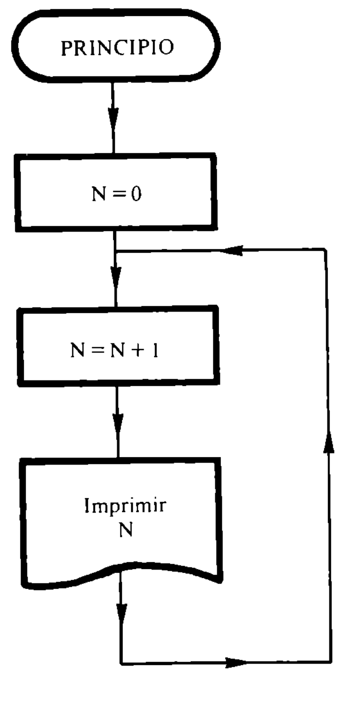


Diagrama de flujo de CONTADOR

Verás que escribe, a toda velocidad, los números:

```
1
2
3
4
5
.
.
.
```

y que no para.

Cada vez que la ejecución pasa por la línea:

```
30 LET N=N+1
```

la variable N aumenta una unidad. A lo largo del proceso, N va tomando los valores 1,2,3,4,...; es como si estuviera contando; por eso decimos que N es un *contador.*

Como has observado, la ejecución del programa no termina nunca. Para detenerla tienes que pulsar la tecla [STOP] o la [BREAK], según la clase de ordenador que utilices. Si, una vez parado el programa, deseas que continúe, tienes que teclear la palabra CONT, y pulsar la tecla [RETURN], naturalmente.

Interrumpe la ejecución pulsando la tecla correspondiente. ¿Qué tal? ¿Se te ocurre un juego? Intenta parar la ejecución justamente cuando el ordenador haya escrito el número 100. Verás que no es fácil: unas veces no llegas y otras te pasas. Puedes jugar con tus amigos. Ganará el juego el que más cerca se quede de 100.

10.4 MAL TIEMPO

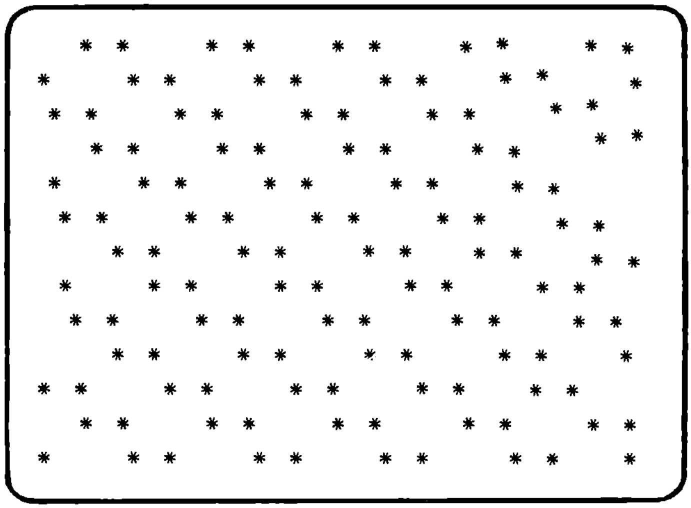
Cuando la pantalla está completamente llena y el ordenador tiene que seguir escribiendo, entonces todas las líneas ya escritas se corren hacia arriba para dejar sitio por abajo a las nuevas impresiones. De esta forma, las líneas de arriba se van perdiendo y las de abajo van subiendo.

Estos movimientos se pueden aprovechar para programar algunos efectos especiales. Pasa el siguiente programa:

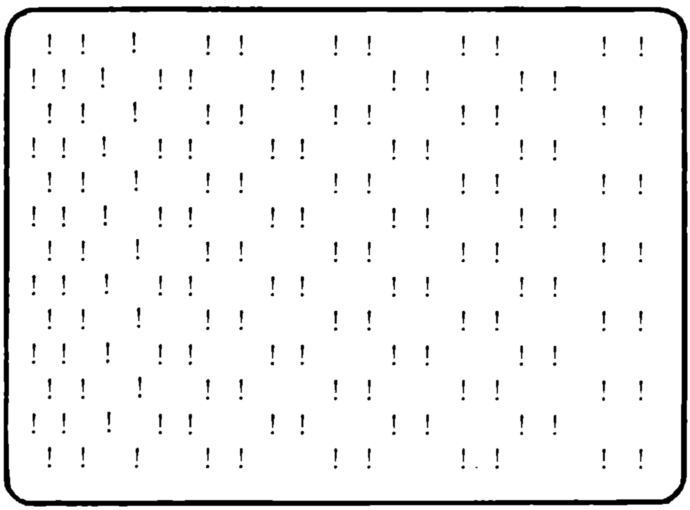
```
10 REM COPOS DE NIEVE
20 PRINT "*  *       ";
30 GOTO 20
40 END
```

La cadena de carácteres de la línea 20 lleva: asterisco (*), 2 espacios, asterisco (*), 7 espacios.

Cuando ejecutes el programa verás desencadenarse una terrible nevada:

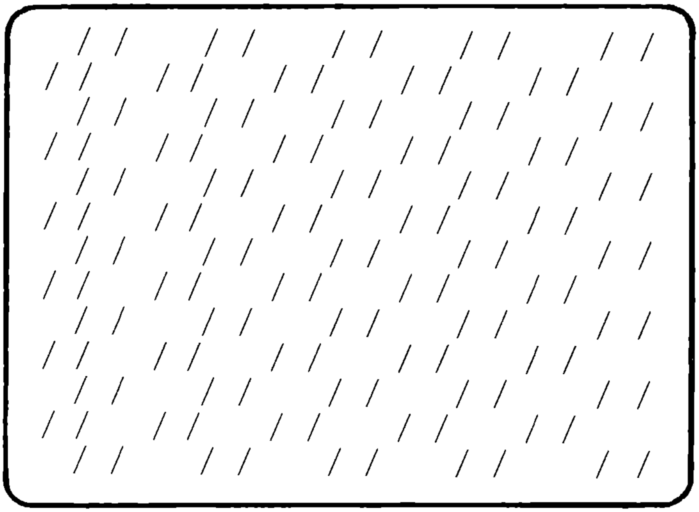


Cambia el asterisco (*) por el signo de admiración (!), y obtendrás una lluvia torrencial:



Te parecerá que "llueve para arriba", pero no todo se puede conseguir con un ordenador.

Si ahora cambias el signo de admiración por la barra inclinada (/), verás el efecto que produce el viento del este sobre la lluvia:



EJERCICIOS

10.1 Mira a ver qué efectos producen estos programas:

```
10 PRINT "*"
20 GOTO 10
```

```
10 PRINT "*";
20 GOTO 10
```

```
10 PRINT " *"
20 GOTO 10
```

```
10 PRINT " *";
20 GOTO 10
```

10.2 Llena toda la pantalla de letras A.

10.3 Haz un programa que escriba muchas veces:

SOY BUEN PROGRAMADOR

10.4 Haz un programa para que el ordenador escriba números pares.

11

BIFURCACIONES

11.1 IF...THEN...

Ya has comprobado lo difícil que resulta parar la ejecución del programa CONTADOR (10.3) justo en el número 100. Sin embargo tu ordenador puede hacerlo, ¡y sin fallar ni una sola vez! Mira cómo:

```
10 REM CUENTO HASTA 100
20 LET N=0
30 LET N=N+1
40 PRINT N
50 IF N=100 THEN END
60 GOTO 30
```

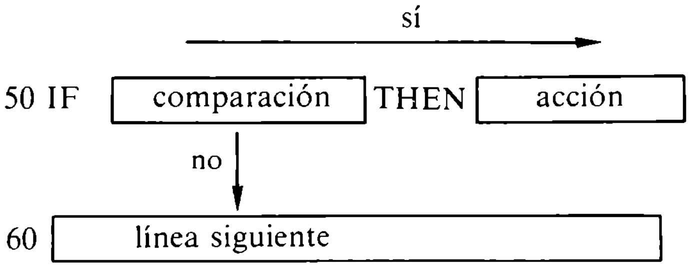
Pruébalo. Verás que se detiene justo después de escribir el 100. Ello es debido a que la línea

```
50 IF N=100 THEN END
```

lleva una instrucción del tipo IF...THEN... (en castellano SI...ENTONCES...), que actúa de la siguiente manera:

Si el valor de la variable N es 100, entonces el ordenador hace lo que está puesto después de la palabra THEN, en este caso END, fin. Cuando el valor de N no es 100, ejecuta la línea siguiente, en nuestro caso la 60 (desde donde vuelve a la 30).

En general la estructura de la instrucción IF...THEN... es la siguiente:



La comparación que figura entre IF y THEN suele ser una igualdad (por ejemplo: N=100), o una desigualdad (por ejemplo: A<B). Si la comparación es verdadera, se ejecuta la *acción* que viene después de THEN. Cuando la comparación no es verdadera, se pasa a la línea siguiente y la ejecución continúa por ahí.

Para formar las comparaciones puedes utilizar los símbolos = ,<, > , con el siguiente significado:

= igual a

< menor que

> mayor que

<= menor o igual que

>= mayor o igual que

<> desigual a (distinto de)

Ahora podemos escribir el programa anterior de otra manera:

```
10 REM CUENTO HASTA 100
20 LET N=0
30 LET N=N+1
40 PRINT N
50 IF N<100 THEN GOTO 30
60 END
```

La línea 50, mientras N sea menor que 100, hace que la ejecución salte a la línea 30. Cuando N es 100, la comparación N<100 es falsa, y la ejecución sigue en la línea siguiente, la 60, donde concluye.

Casi todos los ordenadores admiten que se prescinda de la instrucción GOTO dentro de las sentencias IF-THEN. Puedes escribir sencillamente:

```
50 IF N<100 THEN 30
```

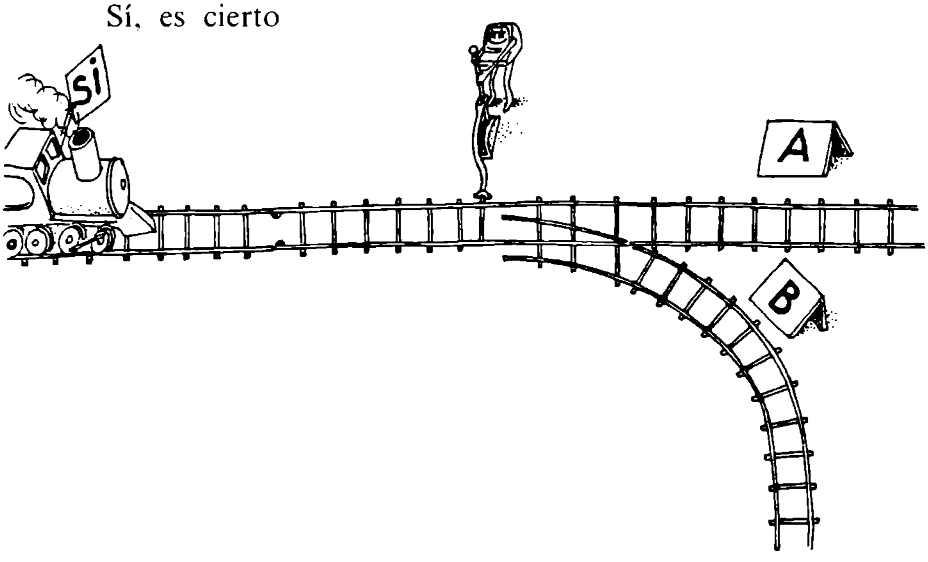
En adelante nosotros haremos uso de esta simplificación. Comprueba si tu ordenador la admite.

11.2 EL GUARDAGUJAS ROBOTIN

Puedes imaginarte la instrucción IF...THEN... como un cambio de agujas en la vía del tren:

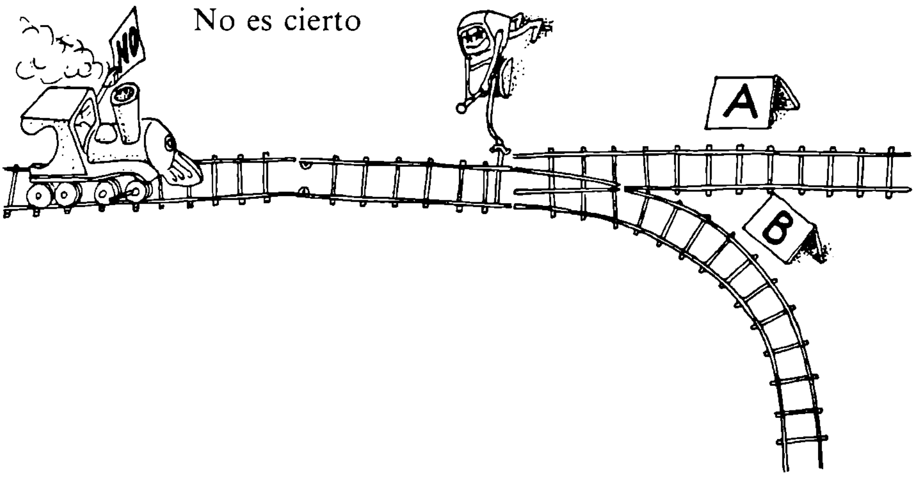
SI | la palanca está hacia atrás |, ENTONCES | sale por A |

Sí, es cierto



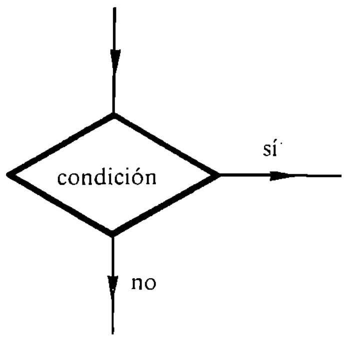
Como se cumple la condición, el tren sale por A.

SI [la palanca está hacia atrás], ENTONCES [sale por A]



Como no se cumple la condición, el tren no puede salir por A, tiene que salir por B.

En los diagramas de flujo se reserva el rombo para indicar las *bifurcaciones* (instrucciones IF...THEN...):



Hagamos, como ejemplo de aplicación, un diagrama que imite el cambio de agujas, de modo que cuando pongamos la palanca en posición ATRAS, el tren salga por la vía A, y cuando la pongamos en posición ADELANTE, salga por la vía B.

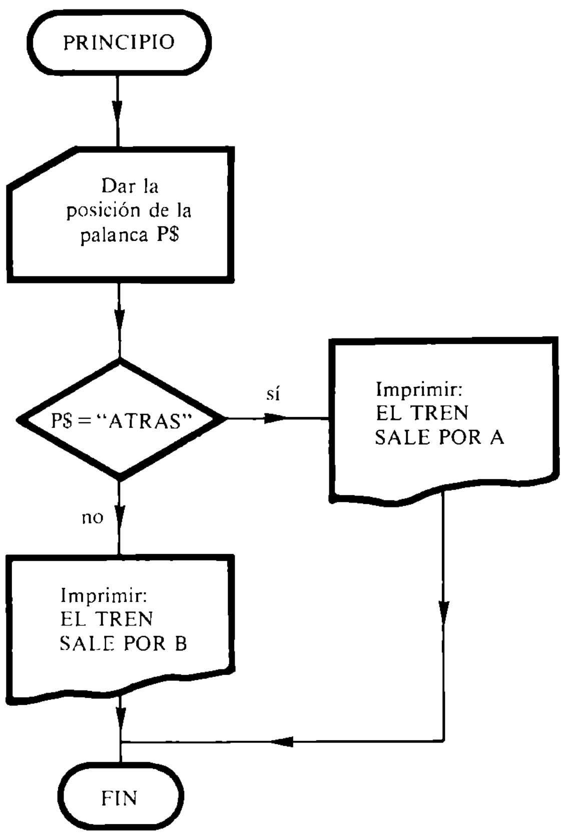


Diagrama de flujo de
CAMBIO DE AGUJAS

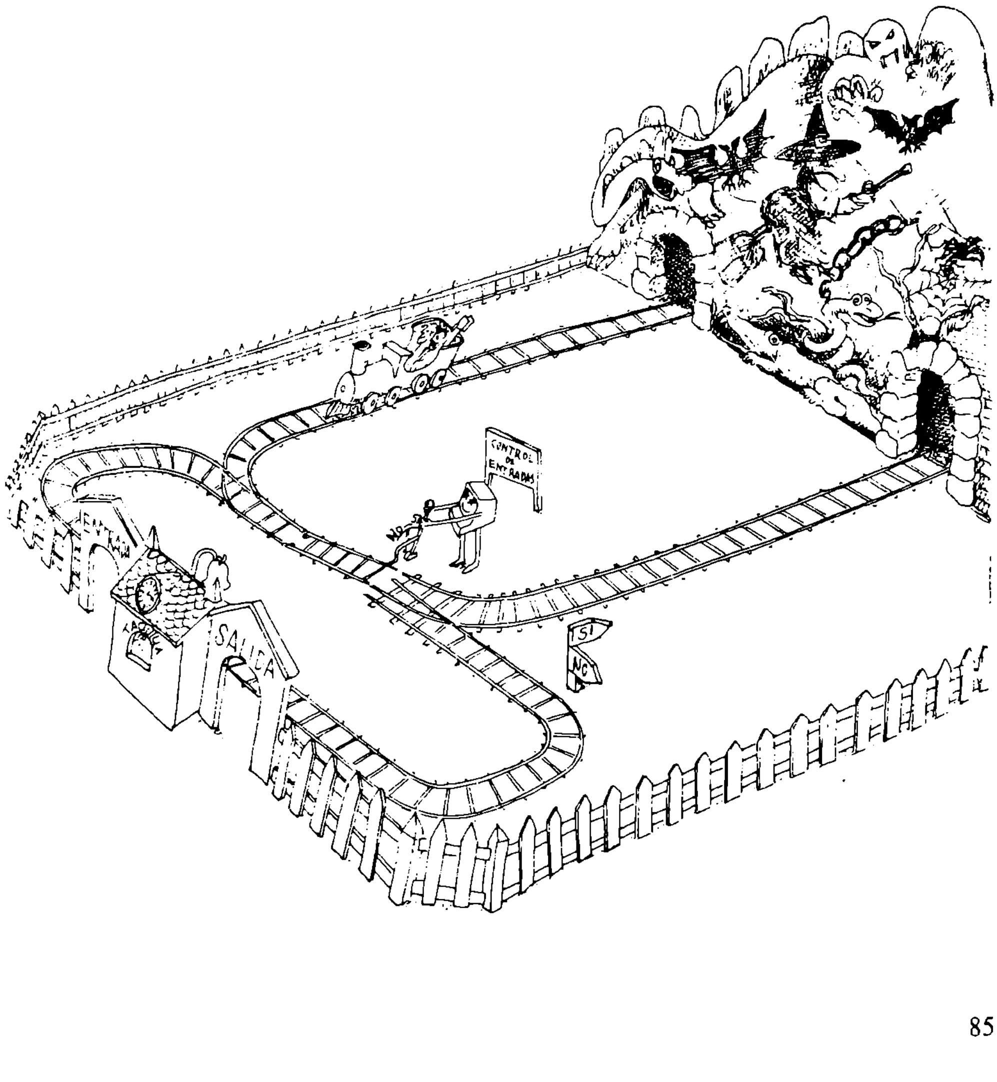
Este programa sirve para imitar el cambio de agujas:

```
10 REM CAMBIO DE AGUJAS
20 INPUT "POSICION DE LA PALANCA";P$
30 IF P$="ATRAS" THEN PRINT "EL TREN SALE POR A"
   :GOTO 50
40 PRINT "EL TREN SALE POR B"
50 END
```

11.3 EL TREN DE LA BRUJA

Esta tarde vas a ir al Parque de Atracciones a montar en el Tren de la Bruja.

En la taquilla compras un número determinado de fichas (F fichas). Nuestro amigo Robotín, que ahora trabaja de portero, controla la entrada al circuito. Si no llevas fichas (F=0), te manda a la calle. Si por el contrario llevas alguna, te deja pasar, te recoge una ficha, y das una vuelta, para pasar de nuevo por el control, y así sucesivamente hasta que no te quede ninguna ficha.



Hagamos el diagrama de flujo del Tren de la Bruja:

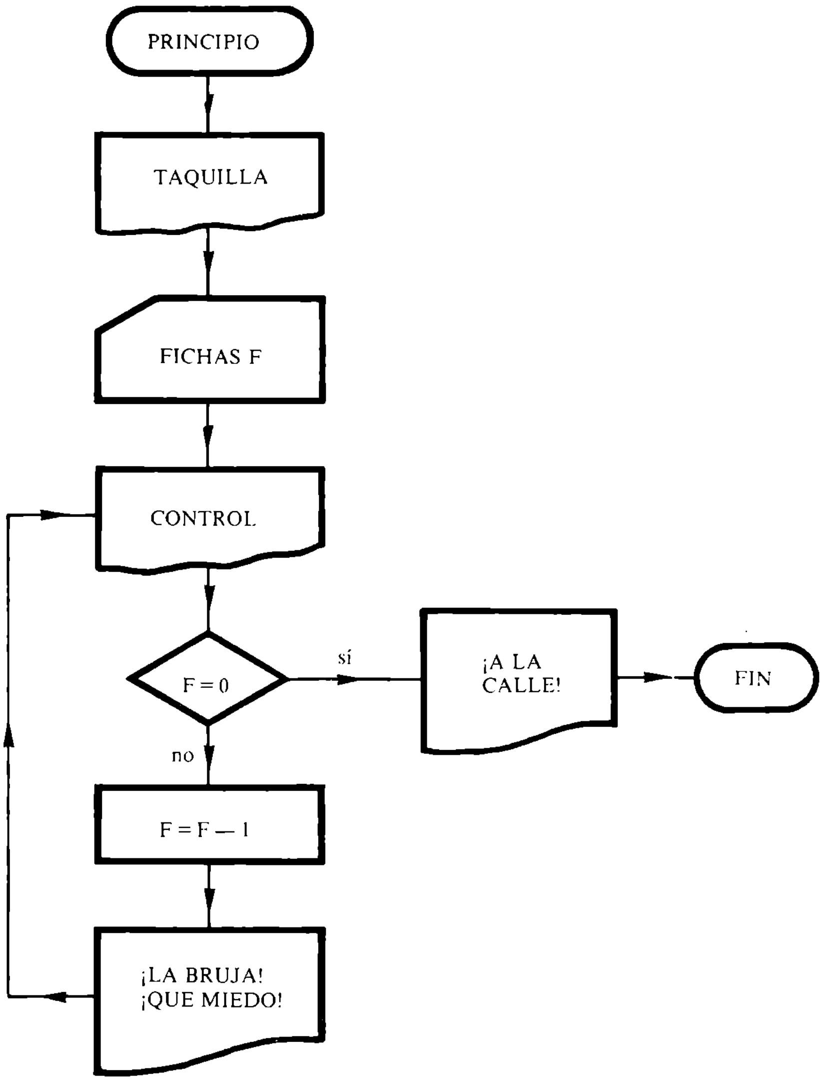


Diagrama de flujo del TREN DE LA BRUJA

Y el correspondiente programa:

```
10 REM TREN DE LA BRUJA
20 PRINT "TAQUILLA"
30 INPUT "CUANTAS FICHAS COMPRAS";F
40 PRINT "CONTROL"
50 IF F=0 THEN PRINT "A LA CALLE":END
60 LET F=F-1
70 PRINT "LA BRUJA! QUE MIEDO!"
80 GOTO 40
```

EJERCICIOS

11.1 Prepara un programa que escriba los números pares, de modo que al llegar al 50 se detenga y escriba YA ACABE.

11.2 Para impedir que personas no autorizadas utilicen tus programas, puedes incluir al principio de ellos unas líneas que pidan al usuario una clave secreta. Si éste da la clave correcta, el programa continúa. Si no, saca el mensaje: LO SIENTO. NO ESTA AUTORIZADO, y el programa termina.

Programa estas primeras líneas con la palabra clave ABRACADABRA.

11.3 Una empresa clasifica a su personal según la estatura. Un empleado es BAJO cuando mide menos de 160 cm., NORMAL cuando su estatura va desde 160 hasta 175 cm., y ALTO cuando mide más de 175 cm. La empresa necesita preparar a su ordenador para que acepte como dato la estatura y escriba BAJO, NORMAL o ALTO, según corresponda. ¿Quieres hacer el programa?

11.4 Este programa sirve para descomponer un número N en sus factores primos. Pruébalo en tu ordenador. Haz el diagrama de flujo que le corresponde.

```
10 REM FACTORES PRIMOS
20 PRINT "CUAL ES EL NUMERO";
30 INPUT N
40 LET D=1
50 LET D=D+1
60 IF D*D >N THEN 120
70 LET R=N-D*INT (N/D)
80 IF R<>0 THEN 50
90 PRINT N,D
100 LET N=N/D
110 GOTO 60
120 PRINT N,N
130 PRINT 1
140 END
```

12

LA SUERTE

12.1 ADIVINA ADIVINANZA

Alguna vez habrás jugado a ese juego que consiste en acertar un número pensado por otra persona. Ahora vas a tener la oportunidad de adivinar el número que previamente "ha pensado" tu ordenador. Pero antes tenemos que programar el juego (en realidad, el verdadero juego es programar). El desarrollo del pasatiempo puede ser más o menos éste:

Tu amigo Robotín "piensa" un número del 1 al 100 y tú escribes el tuyo en la pantalla. El ordenador compara tu número con el suyo, e inmediatamente te da pistas escribiendo en la pantalla: TE HAS PASADO, si tu número es más grande que el suyo; NO HAS LLEGADO, si tu número es más pequeño; o bien, ENHORABUENA. HAS ACERTADO.

Este puede ser el programa:

```
10 REM ADIVINA
20 LET X=47
30 PRINT "DAME UN NUMERO";
40 INPUT N
50 IF N>X THEN PRINT "TE HAS PASADO":GOTO 30
60 IF N<X THEN PRINT "NO HAS LLEGADO":GOTO 30
70 PRINT "ENHORABUENA: HAS ACERTADO"
80 END
```

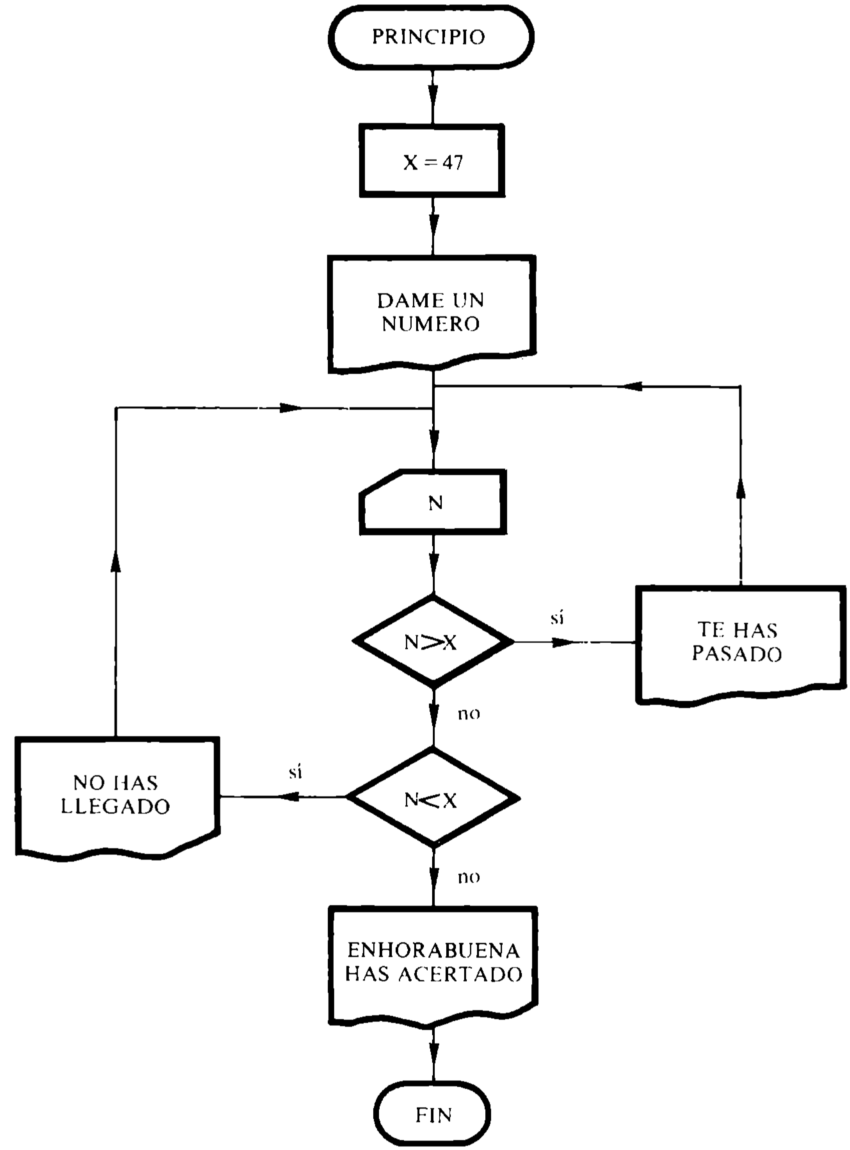


Diagrama del flujo de ADIVINA

Recuerda que en una misma línea de programa se puede poner más de una instrucción, siempre que las separemos con dos puntos (:). Así lo hemos hecho en las líneas 50 y 60; cada una de ellas contiene dos instrucciones: una instrucción IF-THEN y otra GOTO.

Pero ¡vaya un juego! Ya se sabe que el número secreto es el 47. ¡Está escrito en la línea 20!

Puedes hacer una de estas cosas:

1. Hacer como que no lo sabes, y probar el programa.

2. Hacer que juegue un amigo que desconozca el número secreto.

3. Hacer que un amigo cambie la línea 20 sin que lo veas, y jugar tú.

4. Esperar a leer los apartados que siguen, en donde hay recursos con los que puedes programar al ordenador de modo que sea él quien escoja el número secreto.

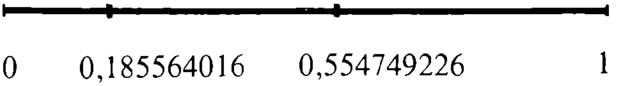
12.2 NUMEROS ALEATORIOS (NUMEROS AL AZAR)

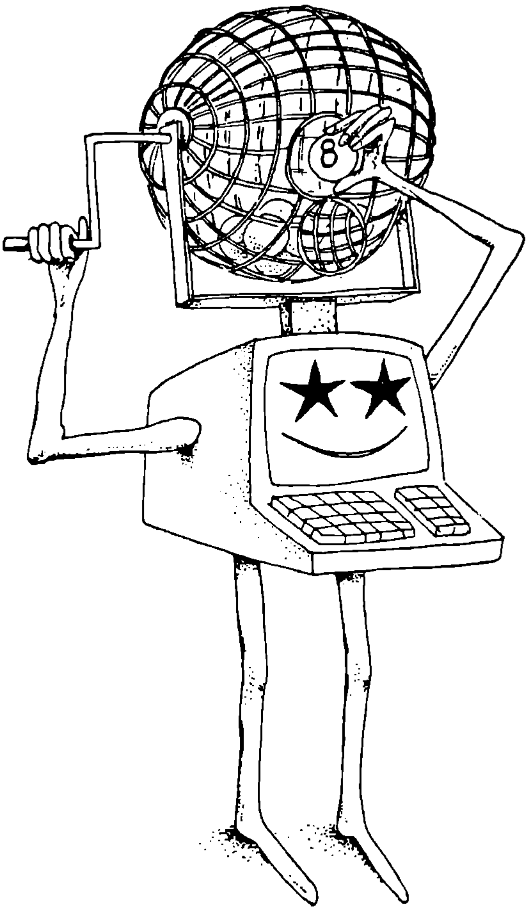
En el juego anterior nos hubiera interesado disponer de un procedimiento para obtener, a suertes, un número comprendido entre 1 y 100.

Nosotros, los humanos, solemos hacerlo de la siguiente manera: escribimos los números en papelitos, los metemos en una bolsa y sacamos uno de ellos sin mirar.

Así no lo puede hacer Robotín. Sin embargo, sí que puede sacar a suertes números decimales comprendidos entre 0 y 1, tales como:

0,185564016; 0,554749226. etc.





El BASIC dispone de una función llamada RND, abreviatura de la palabra inglesa RANDOM (suerte, azar), para obtener números decimales al azar. Estos números son llamados *aleatorios.*

Prueba uno de estos programas, según el ordenador que tengas:

```
10 RANDOMIZE            10 PRINT RND(1)
20 PRINT RND
```

El resultado debe ser un número decimal comprendido entre 0 y 1, que nunca será ni 0, ni 1.

Ejecuta el programa varias veces, y verás que cada vez sale un número aleatorio distinto:

```
RUN
 .897233831
RUN
 .572916244
RUN
 .838893164
RUN
 .931229627
```

Estos números decimales aleatorios no son muy interesantes por sí mismos; pero a partir de ellos el ordenador puede obtener otros que nos interesen.

12.3 APLICACION AL PROGRAMA ADIVINA

¿Cómo podemos sacar números del 1 al 100, a partir de números decimales aleatorios comprendidos entre 0 y 1?

Veamos:

El número RND(1) es mayor que 0 y menor que 1, sin ser 0 ni 1. $0 < RND(1) < 1$

Multiplicando por 100, el resultado 100*RND(1) estará comprendido entre 0 y 100. $0 < 100*RND(1) < 100$

Obteniendo la parte entera, el resultado será alguno de estos números enteros: $0 \leq INT(100*RND(1)) \leq 99$

0, 1, 2, 3, ..., 99

Sumando 1, tenemos alguno de estos números: $1 \leq INT(100*RND(1)) + 1 \leq 100$

1, 2, 3, 4, ..., 100

Ahora sabemos que la línea:

```
20 LET X=INT(100*RND(1))+1
```

produce números enteros aleatorios del 1 al 100, que se almacenan en la variable X.

Esta es, precisamente, la línea que tienes que incluir en el programa ADIVINA, para que Robotín "piense" un número distinto cada vez que se ejecute el programa.

Si tu ordenador no funciona con esa línea, cámbiala por estas dos:

```
15 RANDOMIZE
20 LET X=INT(100*RND)+1
```

Fíjate bien en las fórmulas que hemos utilizado para generar números aleatorios, pues tendrás que utilizarlas con frecuencia.

EJERCICIOS

12.1 Haz un programa que imite una tirada de un dado.

12.2 Diseña un programa que imite el sorteo del cupón pro-ciegos. Tienes que obtener a suertes un número del 0 al 9999.

12.3 Haz un programa que imite el lanzamiento de una moneda. Para ello tendrás que generar los números aleatorios 0 y 1 con la instrucción:

```
LET A=INT(2*RND(1))
```

Luego tendrás que hacer corresponder al 0 la cara y al 1 la cruz.

12.4 Prepara un programa para obtener al azar uno de los resultados 1, X, 2 de una quiniela.

13

AND Y OR, PORTEROS AUTOMATICOS

13.1 PORTERO RIGIDO

Ahora Robotín trabaja de portero automático electrónico. Para dejarte pasar por la puerta, te pide dos contraseñas:

Primera contraseña: PLATA
Segunda contraseña: ORO

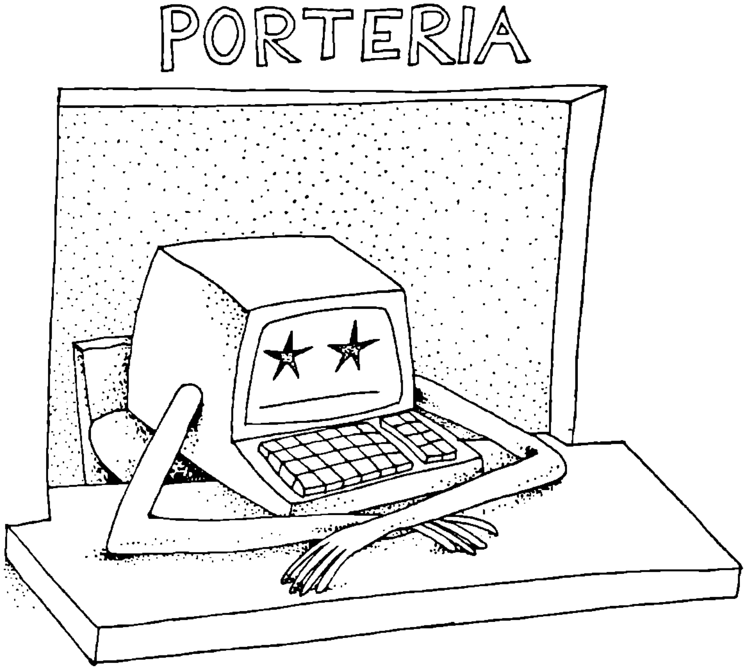
Supongamos que para poder pasar hubiera que conocer *las dos* contraseñas. Este sería el caso de un *portero automático rígido.*

El programa que va a continuación imita al portero rígido. Las variables de cadena PC$ y SC$ recogen, respectivamente, la primera y segunda contraseña que se dan. En la línea 40 se comparan estas contraseñas con las auténticas:

```
10 REM PORTERO AUTOMATICO RIGIDO
20 INPUT "PRIMERA CONTRASEÑA";PC$
30 INPUT "SEGUNDA CONTRASEÑA";SC$
40 IF PC$="PLATA" AND SC$="ORO" THEN PRINT "PASA"
   :GOTO 60
50 PRINT "LO SIENTO. NO PUEDES PASAR"
60 END
```

Fíjate en la línea 40. En ella hay dos condiciones simples: PC$ = "PLATA" y SC$ = "ORO", que unidas mediante la conjunción inglesa AND (y), forman una condición más compleja:

```
PC$="PLATA" AND SC$="ORO"
```



Esta condición se verifica, únicamente, cuando las dos condiciones simples se cumplen. En los restantes casos no se verifica.

El visitante no pasa, si falla una de las dos contraseñas, o las dos. Ejecuta el programa con las distintas posibilidades y comprueba la siguiente tabla:

PORTERO AUTOMATICO RIGIDO		
Primera contraseña	Segunda contraseña	Consecuencia
PLATA	ORO	PASA
PLATA	COBRE	NO PASA
NIQUEL	ORO	NO PASA
BRONCE	ESTAÑO	NO PASA

13.2 PORTERO FLEXIBLE

Un portero automático flexible sería aquél que dejara pasar también cuando sólo acertaras una de las dos contraseñas.

Es muy fácil convertir el portero automático rígido en flexible: basta cambiar, en la línea 40, la palabra inglesa AND por la palabra OR, que significa "o". La nueva línea queda así:

```
40 IF PC$="PLATA" OR SC$="ORO" THEN PRINT "PASA"
   :GOTO 60
```

Comprueba que ahora el portero se comporta así:

PORTERO AUTOMATICO FLEXIBLE		
Primera contraseña	Segunda contraseña	Consecuencia
PLATA	ORO	PASA
PLATA	COBRE	PASA
NIQUEL	ORO	PASA
BRONCE	ESTAÑO	NO PASA

EJERCICIOS

13.1 El ordenador "piensa" una flor (AMAPOLA) y una fruta (PERA). Un amigo tuyo tiene que adivinarle los dos nombres. Prepara un programa para ello.

13.2 Se quiere formar el Club de Amigos de BASIC JUNIOR, para chicos de 10 a 15 años. Haz un programa para que el ordenador admita (o no) a los socios según su edad.

13.3 Los ordenadores también tienen su corazoncito. El tuyo está enamorado de Pilar. Prepárale un programa para que pregunte:

QUIEN ERES?

Y conteste TE AMO, cuando el nombre escrito sea PILAR o PILI. En otros casos que conteste con un frío HOLA.

13.4 Hay rebajas en una tienda de tejidos. Venden los pares de calcetines a 150 ptas., las bufandas a 800 ptas. y los jerseys a 3100 ptas. Si la compra es de más de 5000 ptas., hacen un 20 por 100 de descuento. También lo hacen si te llevas 3 o más prendas. Debes saber que aunque te lleves muchas prendas, y tu factura sea de mucho dinero, el descuento seguirá siendo del 20 por 100, no del 40 por 100.

La cajera dispone de un ordenador en el que introduce el número de artículos de cada clase que has comprado, e inmediatamente aparece en la pantalla el precio total. Diseña un programa que haga esto.

14

LA INSTRUCCION GET (INKEY$)

14.1 METER DATOS SIN PARAR EL PROGRAMA

El BASIC dispone de una instrucción GET parecida a la instrucción INPUT, pero que se diferencia de ésta fundamentalmente en tres cosas:

1. No espera.
2. Admite únicamente un carácter.
3. No saca interrogación en la pantalla.

La instrucción GET A$ mete en la variable A$ la letra o carácter que pulsemos en el instante justo en que el ordenador pasa por ella. Como es casi imposible hacerlo en ese instante, la instrucción GET va frecuentemente acompañada de una instrucción IF...THEN... de la siguiente forma:

```
20 GET A$
30 IF A$="" THEN 20
```

Para comprenderlo mejor, escribe y rueda el siguiente programa:

```
10 REM GET
20 GET A$
30 IF A$="" THEN 20
40 PRINT A$
50 GOTO 20
```

Cuando, al ejecutar el programa, el ordenador encuentra la instrución GET, repasa el teclado rápidamente a ver si tú has pulsado alguna

tecla. Si lo has hecho, guarda el valor de esa tecla en A$; si no, A$ será la cadena nula "", es decir, no tendrá nada y volverá a la línea 20.

También se puede poner GET A, y entonces sólo podrás teclear números y no letras.

En algunos ordenadores la instrucción GET tiene otro nombre, se llama INKEY$. Se utiliza de forma parecida; en nuestro programa habría que sustituir la línea 20 por:

```
20 LET A$=INKEY$
```

14.2 TONTO EL QUE LO LEA

Vas a dar una sorpresa a un amigo tuyo. Mete el siguiente programa en tu ordenador y ejecútalo.

```
10 REM SORPRESA
20 (BORRAR LA PANTALLA) (Ver fin del capítulo)
30 GET A$
40 IF A$="" THEN 30
50 PRINT "TONTO EL QUE LO LEA"
60 END
```

Al rodar el programa se queda la pantalla en blanco. Si no se pulsa una tecla, GET A$ toma la cadena nula y la línea 40 envía a la 30. El programa se estanca en las líneas 30 y 40 con la pantalla en blanco.

Ahora avisa a un amigo tuyo, llévale al ordenador y anímale a manejarlo. En cuanto ponga la mano encima de una tecla, A$ tomará ese valor, deja de ser la cadena nula; la condición de la línea 40 no se cumple, y sale en la pantalla:

```
TONTO EL QUE LO LEA
READY.
■
```

Nota: Todos los microordenadores tienen una instrucción para *borrar la pantalla* desde dentro de un programa. Una versión muy extendida es la instrucción CLS, contracción de CLEAR SCREEN. Otra marca conocida tiene una tecla con la leyenda:

CLR HOME

Esta tecla, apretando a la vez la tecla SHIFT, borra la pantalla desde el teclado, en modo comando. Para borrar la pantalla desde el programa basta poner:

```
20 PRINT  " [CLR HOME] "
```

(apretando Shift a la vez)

Comprobarás que en este caso, en vez de borrar la pantalla, aparece entre las comillas del PRINT un corazoncito. Esa es la señal de que cuando el ordenador ejecute la línea 20, borrará la pantalla.

EJERCICIOS

14.1 Analiza este programa que sirve para medir tus reflejos. Inmediatamente después de ponerlo en marcha tienes que pulsar la tecla [*].

```
10 REM REFLEJOS
20 (BORRAR LA PANTALLA)
30 PRINT TAB (20);"*"
40 GET A$:IF A$< >"*" THEN 30
50 END
```

14.2 Analiza primero, y ejecuta después, este programa:

```
10 REM USO DEL GET
20 PRINT "ESTOY ESPERANDO A QUE ACABES DE
   LEER ESTO"
30 PRINT "PARA CONTINUAR PULSA UNA TECLA"
40 GET A$:IF A$="" THEN 40
50 PRINT "CONTINUO"
60 END
```

14.3 Pasa el programa siguiente y observa lo que ocurre cuando tocas una tecla cualquiera.

```
10 REM GET
20 GET A$
30 IF A$="" THEN PRINT "MU":GOTO 20
40 PRINT "MUS":GOTO 20
50 END
```

14.4 Haz un programa para que, dando a una tecla numérica, te presente un mensaje que diga: "YA ENTIENDO LA INSTRUCCION GET".

15

FOR-NEXT

15.1 COSAS QUE SE REPITEN

En el apartado 11.1 escribíamos los 100 primeros números con las instrucciones IF-THEN y GOTO. El BASIC dispone de un juego de instrucciones que permite resolver este problema fácilmente. Tiene esta forma:

```
   10 REM CONTADOR
┌──20 FOR N=1 TO 100
│  30    PRINT N
└──40 NEXT N
   50 END
```

En castellano diría:

```
┌──DESDE N=1 HASTA 100
│          ESCRIBIR N
└──PASA AL SIGUIENTE N
```

Al ejecutar el programa aparece en la pantalla:

1
2
3
.
.
.
100

Al principio la variable N toma el valor 1 y el ordenador escribe este número; luego N vale 2 y escribe el 2; y así sucesivamente hasta que N vale 100, que es el último valor que se escribe.

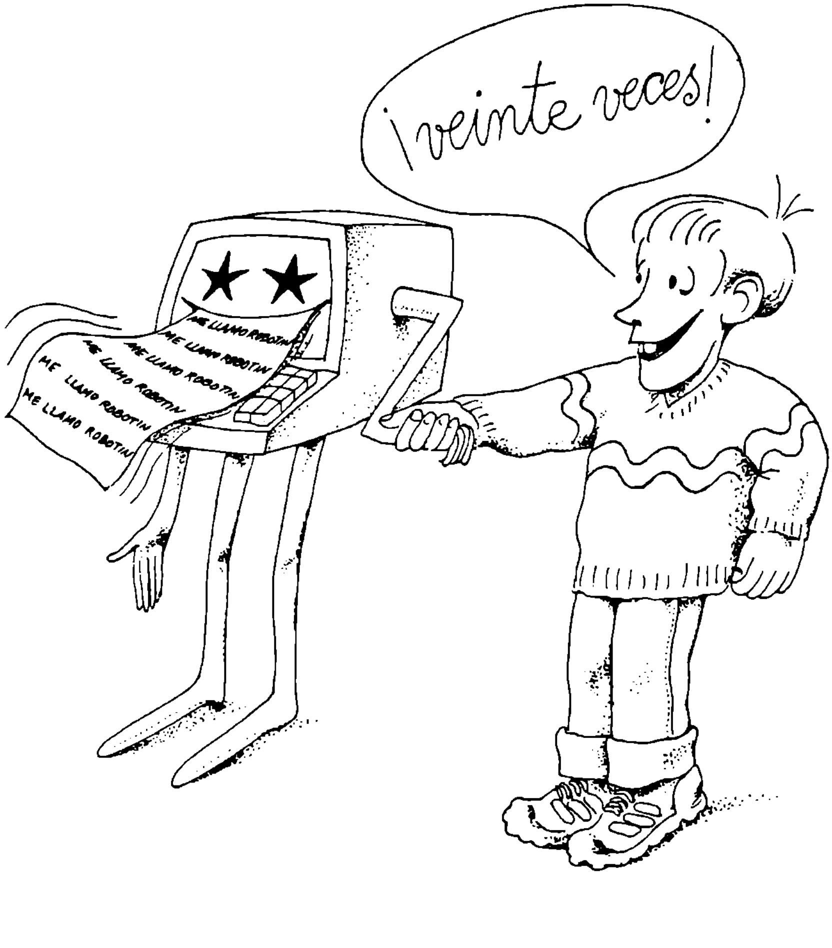
La escritura del ciclo FOR-NEXT presenta siempre este aspecto:

```
┌─FOR I=1 TO 100
│ ┌─────────────────────────────┐
│ │  ACCION QUE SE REPITE       │
│ └─────────────────────────────┘
└─NEXT I
```

La variable I es el *índice*, que va tomando los valores 1, 2, 3, 4,... (en el ejemplo anterior hasta 100). Entre la instrucción FOR y la instrucción NEXT hay un bloque de instrucciones que se repite 100 veces (en este caso).

Suponte ahora que Robotín quiere escribir su nombre 20 veces. Con un ciclo FOR-NEXT es muy sencillo:

```
 10 REM ME LLAMO ROBOTIN
┌20 FOR I=1 TO 20
│30  PRINT "ME LLAMO ROBOTIN"
└40 NEXT I
 50 END
```

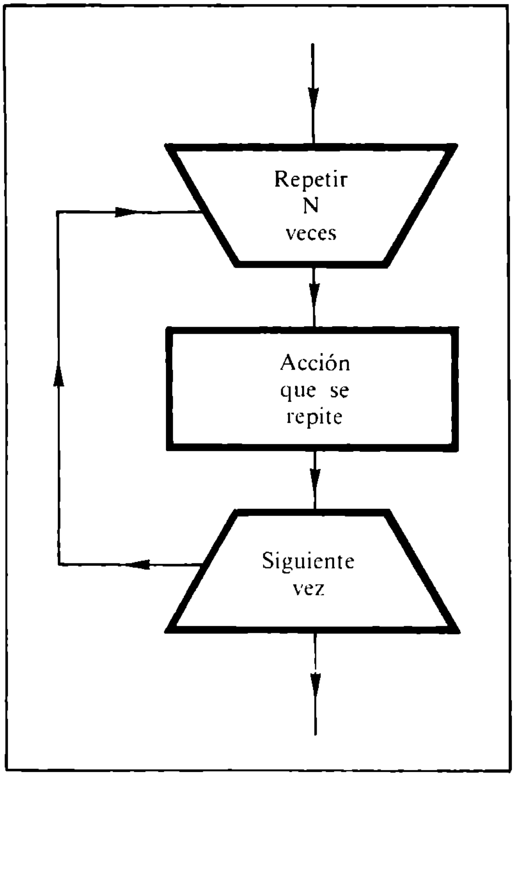


En la pantalla aparecerá:

```
ME LLAMO ROBOTIN
ME LLAMO ROBOTIN
.................
.................
ME LLAMO ROBOTIN
```

(20 veces)

En los diagramas de flujo, el ciclo FOR-NEXT viene representado por:



El organigrama del último programa puede ser:

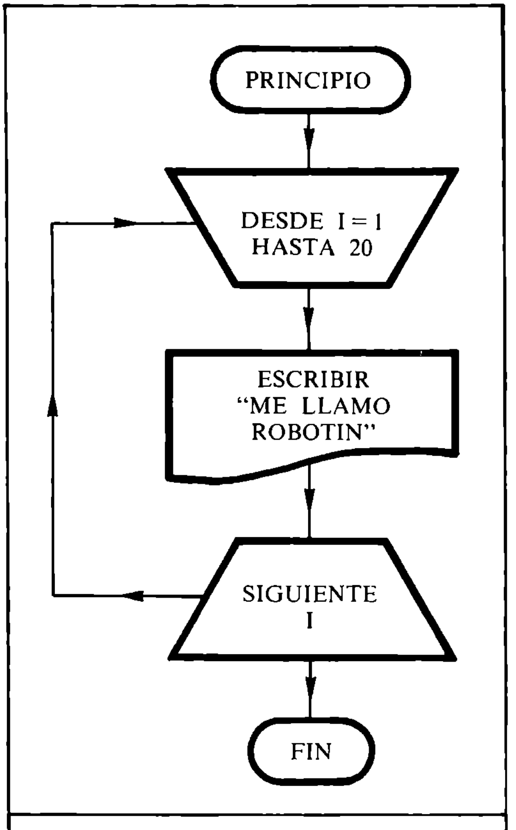


Diagrama de flujo de
ME LLAMO ROBOTIN

15.2 SUMADOR

A finales del siglo XVIII un maestro propuso a sus alumnos el siguiente ejercicio: Sumar los cien primeros números.

$$1 + 2 + 3 + 4 + 5 + 6 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + 100 = S$$

Apenas habían transcurrido unos instantes cuando un niño dio la respuesta correcta: "La suma es 5.050".

Este niño se llamaba Gauss y años después se convertiría en un gran matemático.

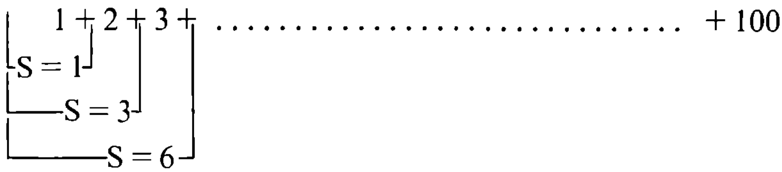
El niño tuvo esta feliz idea:

$$\begin{array}{l}
1+\ 2+\ 3+\ \ldots\ldots\ldots\ +98\ \ +99+100\ \ =S \\
100+99+98+\ \ldots\ldots\ldots\ +\ 3\ \ \ +2\ \ \ +1\ \ =S \\
\hline
101+101+101+\ \ldots\ldots\ +101+101+101=2S
\end{array}$$

$$101\times 100=2S$$

$$S=\frac{101\times 100}{2}=5.050$$

El ordenador lo puede hacer de una forma más tosca, pero muy rápidamente: Sumando uno a uno los 100 números.



Si llamas S a la suma parcial en cada momento, al final del proceso tendrás en S la suma total.

La clave del programa está en la siguiente instrucción que va a repetirse 100 veces:

LET S	=	S	+	N
Suma actual		Suma anterior		Número sumado

Antes de empezar a sumar tienes que poner a cero la variable S: S=0.

```
10 REM SUMADOR
20 LET S=0
30 FOR N=1 TO 100
40  LET S=S+N
50 NEXT N
60 PRINT "LA SUMA ES = ";S
70 END
```

Observa que la ejecución del programa se hace de este modo:

	Suma actual		Suma anterior		Número sumado
Si N = 1	1	=	0	+	1
Si N = 2	3	=	1	+	2
Si N = 3	6	=	3	+	3

y así hasta N = 100.

15.3 STEP

Ahora queremos calcular la suma de los números impares menores que 100.

$$1 + 3 + 5 + 7 + \ldots\ldots\ldots\ldots + 99$$

Este problema es casi igual que el anterior. La única diferencia es que N "va de dos en dos". El ciclo FOR-NEXT está preparado para abordar esta situación.

```
10 REM SUMA DE NUMEROS IMPARES
20 LET S=0
30 FOR N=1 TO 100 STEP 2
40  LET S=S+N
50 NEXT N
60 PRINT "LA SUMA ES = ";S
70 END
```

Hemos incluido, en la instrucción FOR, el STEP 2 (paso o salto). Con esto indicamos que N aumenta de dos en dos.

15.4 TABLERO DE AJEDREZ

Si alguna vez se te ocurre la idea de programar el juego del ajedrez para jugar contra tu ordenador, lo primero que tendrás que hacer es dibujar el tablero.

Para ello puedes seguir varios caminos; uno de ellos es imprimir ocho filas de ocho cuadraditos cada una:

Imprimir fila 1ª: 

Imprimir fila 2ª: 

....

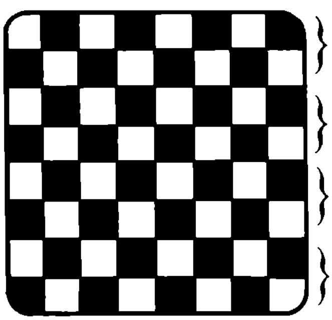
....

....

Imprimir fila 7ª: 

Imprimir fila 8ª: 

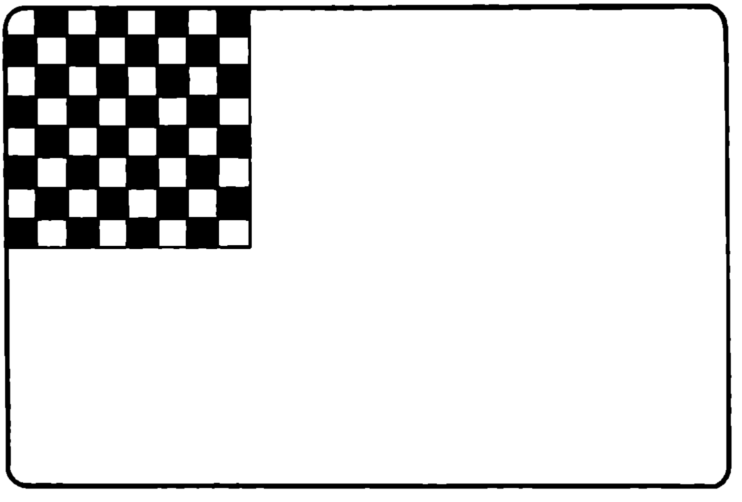
Si observas el tablero, puedes ver que las filas 1ª y 2ª "forman un bloque" que se repite cuatro veces.



Esto aconseja utilizar el ciclo FOR-NEXT.

```
10  REM TABLERO DE AJEDREZ
20 (BORRAR LA PANTALLA)
30 FOR I=1 TO 4
40 PRINT "■□■□■□■□"
50 PRINT "□■□■□■□■"
60 NEXT I
70 END
```

Este programa "pinta" el tablero en el ángulo superior izquierdo.



Retoca el programa para que el tablero aparezca en medio de la pantalla.

EJERCICIOS

15.1 Utiliza ciclos FOR-NEXT para hacer un programa que dibuje esta bandera:

```
x x x x x
x x x x x
x x x x x
x
x
x
x
```

15.2 Castiga a tu ordenador a escribir veinte veces:

VOY A SER BUENO

15.3 Haz un programa que escriba los 50 primeros múltiplos de 3.

15.4 Escribe un programa que te presente en pantalla los números del 20 al 1. Observa que van disminuyendo de uno en uno. (El STEP puede tomar valores negativos).

16

LAS APARIENCIAS ENGAÑAN

16.1 UN CUENTO ORIENTAL

Cuenta la leyenda que un rey de la India, entusiasmado por el juego del ajedrez, quiso conceder a su inventor todo lo que éste deseara.

"Me conformo, Majestad —respondió el inventor—, con que me deis un grano de trigo por la primera casilla, dos por la segunda, cuatro por la tercera, y así hasta completar las sesenta y cuatro casillas".

El rey sonrió con benevolencia y ordenó a su tesorero que cumpliera el encargo.

El tesorero tiene que sumar:

$$1 + 2 + 4 + 8 + 16 + \ldots.$$

y así, hasta la casilla 64.

Para hallar esta suma hay que calcular primero el número de granos, N, que hay en cada casilla.

1	2	4	8	16			

Para ello empezamos haciendo N = 1:

```
LET N=1
```

Y a partir de la casilla 2 hasta la 64:

```
LET N=N*2
```

Por otra parte hay que ir sumando los granos de trigo. La suma empieza valiendo 1:

```
LET S=1
```

y en adelante, hasta la casilla 64, la suma va siendo:

LET S	=	S	+	N
La suma vale		La suma anterior		Más los granos de esta casilla

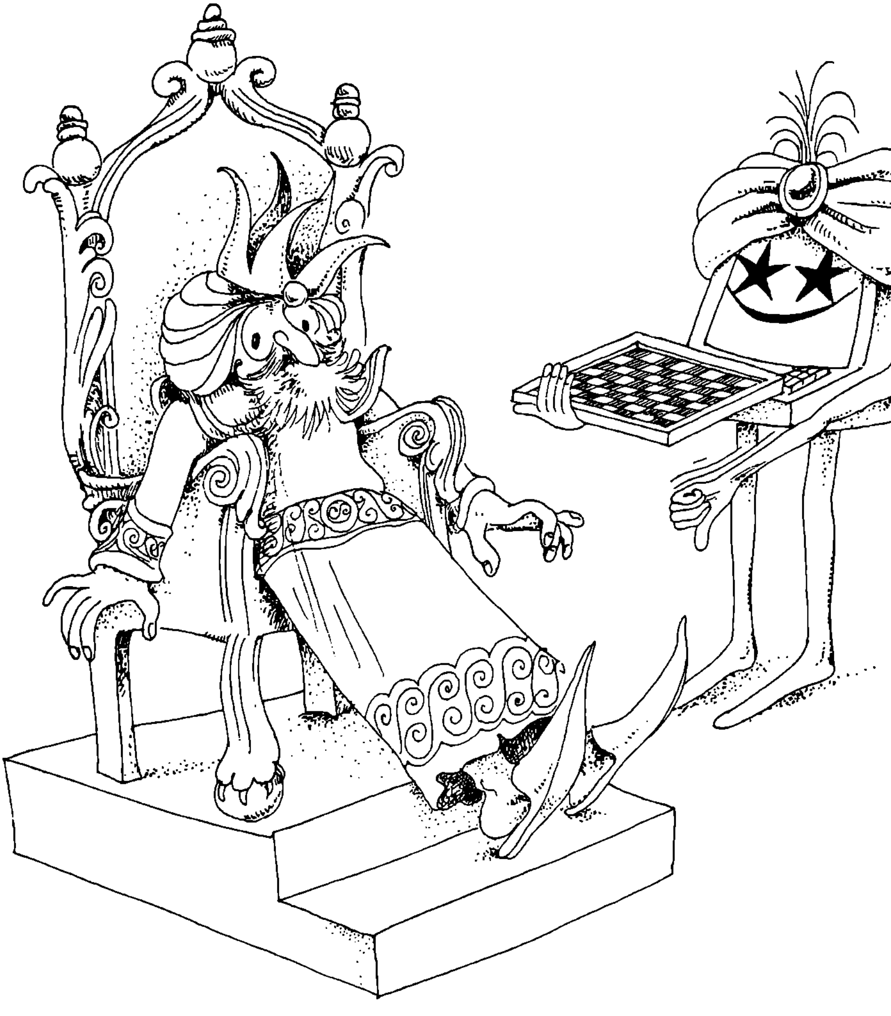
Al final, N contiene el número de granos de la casilla 64 y S la suma de todos los granos del tablero, de principio a fin.

Todo esto se puede resumir en el siguiente algoritmo:

1. Hacer N = 1 y S = 1 (Casilla primera).
2. Hacer 63 veces (Casillas 2 a 64).
   2.1 Calcular el número de granos de la casilla.
   2.2 Sumar este número a la suma anterior.
   Siguiente vez.
3. Escribir la suma.
4. Fin.

que nos conduce al siguiente programa:

```
10 REM LAS APARIENCIAS ENGAÑAN
20 LET N=1:LET S=1:REM PRIMERA CASILLA
30 FOR I=2 TO 64:REM CASILLAS 2 A 64
40 LET N=N*2
50 LET S=S+N
60 NEXT I
70 PRINT "EL TESORERO TIENE QUE ENTREGAR";S;"GRA
   NOS DE TRIGO"
80 END
```



16.2 EL REY QUE RABIO

Si calculamos a mano los granos de trigo, sale la asombrosa cifra de:

18.446.744.073.709.551.615

Tu ordenador posiblemente no puede expresar un número tan grande. Verás en la pantalla un número parecido a:

1.84467441E + 19

Recuerda que es una forma de escribir

$$1{,}84467441 \times 10^{19}$$

que es igual a:

18.446.744.100.000.000.000

Cuando el tesorero informó a Su Majestad de la cantidad que tenía que entregar al astuto inventor, suponemos que debió de cambiar de color.

16.3 LA PRUDENCIA ES UNA VIRTUD

La historia del rey y el inventor del ajedrez tiene un final desastroso para el monarca.

Si el rey hubiera sospechado que la ingenua petición exigía una enorme cantidad de trigo, antes de responder, habría dicho a su tesorero que hiciera los cálculos oportunos para que la recompensa fuera generosa, pero razonable.

El premio podía haber sido el trigo contenido en un almacén de 100 metros de largo, 20 metros de alto y 30 metros de ancho.

Este depósito tiene un volumen de:

$$V = 100\ \text{m} \times 20\ \text{m} \times 30\ \text{m} = 60.000\ \text{m}^3$$

Cada m^3 de trigo contiene por término medio unos 15 millones de granos, lo que da una cantidad de trigo igual a:

$60.000 \ m^3 \times 15.000.000$ granos/$m^3 = 900.000$ millones de granos $= 9 \times 10^{11}$ granos.

Si el tesorero hubiera tenido ordenador, habría introducido una "señal de alarma" en el programa "LAS APARIENCIAS ENGAÑAN", para el caso de que se llegara a superar la cantidad anterior.

```
10 REM SALIR ANTES DE TIEMPO
20 LET N=1:LET S=1
30 FOR I=2 TO 64
40 LET N=N*2
50 LET S=S+N
60 IF S>=9E11 THEN 80
70 NEXT  I
80 PRINT "ATENCION PELIGRO:HAY CONTADOS";S;
90 PRINT "GRANOS HASTA LA CASILLA";I
100 END
```

Los resultados obtenidos habrían confirmado las sospechas del soberano. Obtendremos:

$S = 1.09951163E + 12 = 1.09951163 \times 10^{12} = 1099511630000$ granos hasta la casilla 40.

Como has observado, de un ciclo se puede salir desde cualquier instrucción, pero no se puede entrar a no ser que se haga por la primera, es decir, por la instrucción FOR.

```
┌─FOR I=1  TO N           ┌─FOR I=1  TO N
│ .....................   │ .....................
│┌.....................   │┌→....................
└┼NEXT I                  └┼NEXT I
 │                         │
 ▼ SALIDA CORRECTA         │ ENTRADA INCORRECTA
```

EJERCICIOS

16.1 La línea de programa:

FOR J = 1 TO 5000 : NEXT J

no hace nada, pero entretiene al ordenador recorriendo 5000 veces el ciclo FOR-NEXT, cosa que le lleva algún tiempo.

Este truco puedes emplearlo cuando necesites que la ejecución de un programa se detenga (aparentemente) unos instantes. Utilízalo para hacer un programa que:

1º Borre la pantalla.
2º Escriba "TE ESTOY DEJANDO TIEMPO PARA QUE LEAS ESTE LETRERO".
3º Borre la pantalla.

16.2 Un pobre le dijo a un rico: "Cada día, durante un mes, le pagaré 100.000 ptas.; a cambio Vd. me dará, el primer día una peseta, el segundo día dos pesetas, el tercero cuatro, es decir, cada día doble cantidad que la anterior, y así hasta acabar el mes".

Ante tan ventajosa propuesta, el rico aceptó inmediatamente, frotándose las manos.

Haz un programa que calcule las cantidades entregadas por cada uno, y escriba algún comentario jocoso al que se lo merezca.

16.3 De un ciclo se puede salir sin necesidad de haberlo recorrido todas las veces.

```
10 REM SALIDA DE UN CICLO
20 PRINT "VOY A VER SI CUENTO HASTA 100"
30 FOR I=1 TO 100
40 PRINT I
50
60 NEXT I
70 PRINT "LO SIENTO: NO SE CONTAR MAS ALLA DE 5
   9"
80 END
```

Completa la línea 50 para que el último número que se escriba sea el 59.

16.4 Observa lo que ocurre cuando ejecutas el siguiente programa:

```
10 FOR I=1 TO 5
20 PRINT I*I
30 NEXT I
40 PRINT "OBSERVA LO QUE PASA"
50 GOTO 20
60 END
```

17

EL ORDENADOR PROFESOR

17.1 REPASEMOS GEOGRAFIA

Africa es un continente del que sabemos muy poco, y repasar o aprender nombres de ciudades y capitales no viene mal. Consultando un atlas formamos la tabla:

PAIS	CAPITAL
ARGELIA	ARGEL
LIBIA	TRIPOLI
EGIPTO	EL CAIRO
MAURITANIA	NOUAKCHOTT
NIGER	NIAMEY
NIGERIA	LAGOS
CHAD	N'DJAMENA
SUDAN	JARTUM
SOMALIA	MOGADISCIO
ANGOLA	LUANDA

Para obtener esta tabla en la pantalla hay que realizar:

1. Imprimir las palabras PAIS y CAPITAL en el lugar adecuado
   y una raya debajo.
2. Repetir 10 veces.
   EL PAIS y su CAPITAL.
   Siguiente vez.
3. Fin.

Esto es precisamente lo que hace el siguiente programa:

```
10 REM REPASEMOS GEOGRAFIA
20 REM ENCABEZAMIENTO
30 PRINT TAB(5);"PAIS";TAB(20);"CAPITAL"
40 FOR I=1 TO 39:PRINT "-";:NEXT I
50 PRINT
60 REM TABLA
70 FOR I=1 TO 10
80 READ P$,C$
90 PRINT TAB(5);P$;TAB(20);C$
100 NEXT I
110 DATA ARGELIA,ARGEL,LIBIA,TRIPOLI,EGIPTO,EL-C
    AIRO
120 DATA MAURITANIA,NOUAKCHOTT,NIGER,NIAMEY,NIGE
    RIA,LAGOS
130 DATA CHAD, N'DJAMENA,SUDAN,JARTUM,SOMALIA,MO
    GADISCIO
140 DATA ANGOLA,LUANDA
150 END
```

Mediante el ciclo FOR-NEXT, para I = 1, se leen P$ y C$ que son Argelia y Argel respectivamente, y se imprimen a continuación. Para I = 2, P$ y C$ "toman los valores" Libia y Trípoli, y así sucesivamente hasta agotar las líneas DATA.

Prueba a pasar el programa sin la línea 50. ¿Qué ocurre?

17.2 LA TABLA DE MULTIPLICAR

Para repasar las tablas de multiplicar Robotín te propone el siguiente programa:

```
10 REM TABLA DE MULTIPLICAR
20 INPUT "QUE NUMERO QUIERES REPASAR";N
30 FOR I=1 TO 10
40 PRINT N;"*";I;"=";N*I
50 NEXT I
60 END
```

Si ejecutas el programa aparece en la pantalla:

```
QUE NUMERO QUIERES REPASAR?
```

Puedes contestar cualquier número entero, pero lo razonable es que N sea un número entre 1 y 10; por ejemplo 7. Inmediatamente verás aparecer la tabla:

```
QUE NUMERO QUIERES REPASAR?7

7* 1=7
7* 2=14
7* 3=21
7* 4=28
7* 5=35
7* 6=42
7* 7=49
7* 8=56
7* 9=63
7* 10=70
```

Observa la línea 40. En ella hay signos que van entre comillas y otros que no. Lo que está entrecomillado aparece en la pantalla tal cual, y las variables, que no están entrecomilladas, con el valor que les corresponde.

Por ejemplo, para N = 7 e I = 2 la línea

```
                      PRINT N; "*"; I; "="; N*I
                            ↓    ↓    ↓   ↓     ↓
produce la impresión        7    *    2   =     14
```

EJERCICIOS

17.1 Haz un programa que escriba diez capitales europeas y su distancia a Madrid.

17.2 Prepara un programa que escriba la tabla:

```
1  UNO
2  DOS
3  TRES
.  .
.  .
.  .
10 DIEZ
```

Te sugerimos que emplees estas líneas:

```
100 DATA UNO,DOS,TRES,CUATRO,CINCO
110 DATA SEIS,SIETE,OCHO,NUEVE,DIEZ
```

y que las leas con un ciclo FOR-NEXT, con el que también podrás escribir la tabla.

17.3 Programa la tabla de sumar del 5. Modifica el programa para que salga la tabla de cualquier número que tú le des.

17.4 Haz un programa para calcular la media de una serie de datos. Imagínate que los ocho compañeros de un equipo de trabajo decidieran hacer un fondo común con las notas de matemáticas y después repartirse el fondo a partes iguales. El número que correspondería a cada uno sería la nota media. Supongamos que las notas fueran:

9, 7, 3, 6, 9, 5, 2 y 9.

La suma de las notas es $S=50$

La nota media es $M=\frac{50}{8}=6{,}25.$

En general, la media de una serie de datos es:

$$\text{Media}=\frac{\text{Suma de todos los datos}}{\text{Número de datos}}$$

El programa que tienes que diseñar ha de valer para cualquier número de datos, no exclusivamente para el caso de ocho, como en el ejemplo. El número de datos puedes meterlo en la variable N con una instrucción INPUT. Los datos los metes uno a uno en la variable D, que los acumula en la viariable S, en donde van quedando las sumas parciales. Al final, en S estará la suma total. El siguiente diagrama te ayudará a construir el programa:

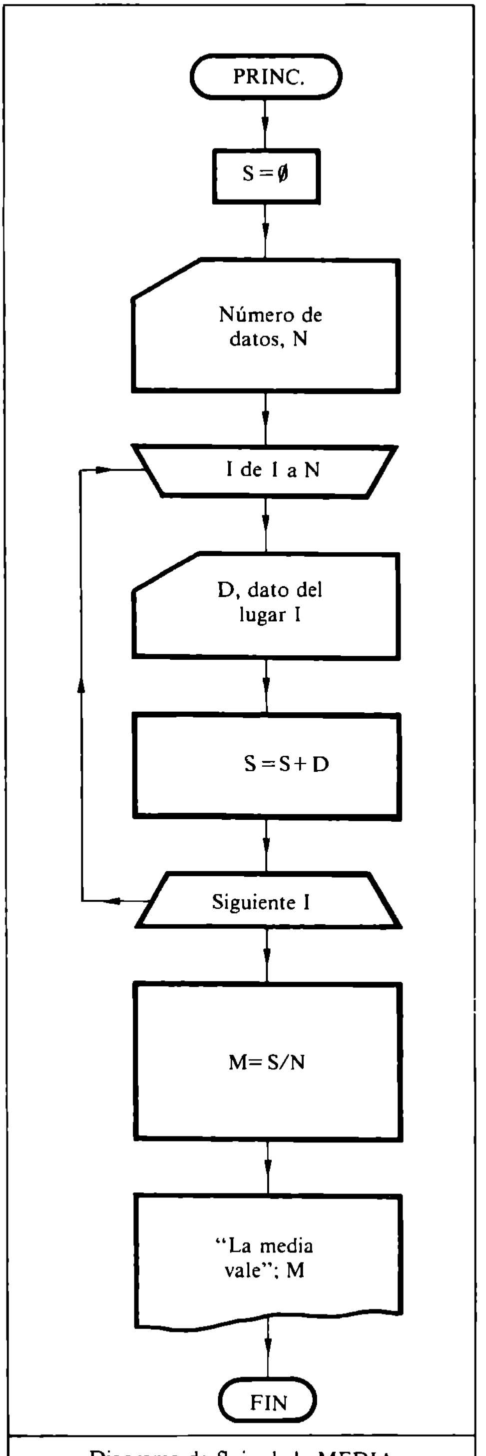


Diagrama de flujo de la MEDIA

18

JUGUEMOS A LOS DADOS

18.1 LANCEMOS UN DADO

Recuerda que en el apartado 12.3 vimos la instrucción:

LET X = INT(100 * RND(1)) + 1

que servía para obtener números enteros aleatorios del 1 al 100. De manera análoga podemos obtener números aleatorios del 1 al 6, como si estuviéramos lanzando un dado. La instrucción

LET D = INT(6 * RND(1))+1

sirve para eso: guarda en la variable D un número del 1 al 6.

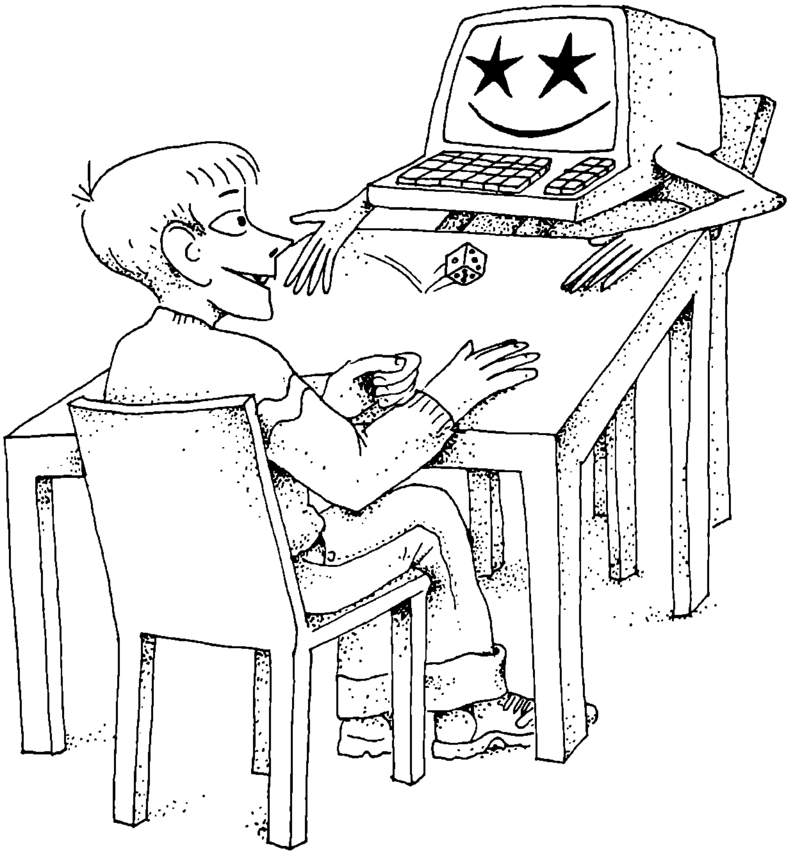
Un programa que imita el lanzamiento de un dado es el siguiente:

```
5 REM LANZAMIENTO DE UN DADO
20 LET D=INT(6*RND(1))+1
30 PRINT D
40 END
```

Ejecútalo varias veces. Verás que imita bastante bien al dado.

Si tu ordenador no puede ejecutar ese programa, cambia la línea 20 por estas dos:

```
10 RANDOMIZE
20 LET D=INT(6*RND)+1
```



18.2 LANCEMOS MIL VECES UN DADO

Si quieres, puedes lanzar el dado mil veces, sin más que añadir al programa un par de líneas para introducir un ciclo FOR-NEXT:

```
5 REM LANZAMIENTO DE UN DADO MIL VECES
15 FOR I=1 TO 1000
20 LET D=INT(6*RND(1))+1
30 PRINT D
35 NEXT I
40 END
```

Cuando pases el programa, verás desfilar al azar por la pantalla los distintos puntos del dado, y quizá sientas la curiosidad de saber si cada

uno de ellos aparece aproximadamente 167 veces, que es, más o menos, la sexta parte de 1000.

Es posible contar cuántas veces aparece cada uno de los números 1, 2, 3, 4, 5 y 6. Para ello tienes que establecer seis *contadores* (variables a los que se les va sumando la unidad):

UN = UN + 1	para contar los unos
DO = DO + 1	para contar los doses
TR = TR + 1	para contar los treses
CU = CU + 1	para contar los cuatros
CI = CI + 1	para contar los cincos
SE = SE + 1	para contar los seises

Suponiendo que en cada tirada la puntuación obtenida se guarde en la variable D, el proceso de recuento se puede organizar así:

Si D=1, entonces UN = UN + 1
Si D=2, entonces DO = DO + 1
Si D=3, entonces TR = TR + 1
Si D=4, entonces CU = CU + 1
Si D=5, entonces CI = CI + 1
Si D=6, entonces SE = SE + 1

Este puede ser el programa:

```
10 REM RECUENTO DE LOS PUNTOS OBTENIDOS
20 REM AL LANZAR UN DADO MIL VECES
30 LET UN=0:LET DO=0:LET TR=0
40 LET CU=0:LET CI=0:LET SE=0
50 FOR I=1 TO 1000
60 LET D=INT(6*RND(1))+1
70 IF D=1 THEN LET UN=UN+1
80 IF D=2 THEN LET DO=DO+1
90 IF D=3 THEN LET TR=TR+1
100 IF D=4 THEN LET CU=CU+1
110 IF D=5 THEN LET CI=CI+1
120 IF D=6 THEN LET SE=SE+1
130 NEXT I
140 PRINT "UNOS:";UN
150 PRINT "DOSES:";DO
160 PRINT "TRESES:";TR
```

```
170 PRINT "CUATROS:";CU
180 PRINT "CINCOS:";CI
190 PRINT "SEISES:";SE
200 END
```

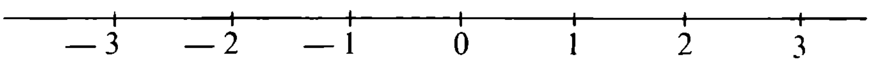
EJERCICIOS

18.1 Forma un número de seis cifras haciendo girar una ruleta de 0 a 9.

18.2 Haz un programa que imite el lanzamiento de una moneda 100 veces, y que indique al final el número de caras y de cruces obtenido.

18.3 El juego consiste en lanzar dos dados. Si la suma es 7 u 11, gana el ordenador, y si no, ganas tú. Programa una partida de diez juegos.

18.4 Tienes una recta en la que están representados los números 0, 1, 2, 3, ... y — 1, — 2, — 3, ... Imagina, como puedas, que en el punto 0 está situada una pulga que va a iniciar una serie de saltos de longitud uno, y que, en cada punto, decide al azar (como si lo echara a cara o cruz) si va a saltar hacia la derecha o hacia la izquierda.



Supón que la pulga da mil saltos y se detiene. ¿Cuál será la posición final? ¿Hasta qué puntos extremos (por la derecha y por la izquierda) habrá llegado en su loco paseo?

Te damos algunas pistas para que te ayuden a construir el programa. La variable P marca la posición de la pulga. Inicialmente P = 0. Las posiciones sucesivas se establecen con la instrucción P = P + S, en donde S es el salto, que puede valer 1 ó — 1, según sea hacia la derecha o hacia la izquierda. El salto se puede calcular así:

$$S = 2 * X - 1$$

donde X es un número aleatorio, 0 ó 1:

X = INT(2 * RND(1))

La posición más a la derecha la puedes guardar en la variable D. Al principio D = 0. En D se llevan los "records" por la derecha, por lo que cambiará de valor cada vez que sea S>D. Será necesaria una línea de tipo:

IF S>D THEN LET D = S.

Análogas consideraciones podríamos hacer para la variable I, que marca la posición extrema izquierda.

19

CICLOS ENCAJADOS

19.1 CICLOS ENCAJADOS

Ya sabes que un ciclo FOR-NEXT tiene esta forma:

```
┌─FOR I=1 TO N
│ ┌─────────────────┐
│ │ACCION QUE SE    │
│ │REPITE N VECES   │
│ └─────────────────┘
└─NEXT I
```

A su vez *la acción que se repite* puede estar formada por otro ciclo FOR-NEXT. Veamos un sencillo ejemplo de este caso:

```
 10 REM CICLOS ENCAJADOS
┌20 FOR I=1 TO 2
│┌30 FOR J=1 TO 3
││40 PRINT I; J
│└50 NEXT J
└60 NEXT I
 70 END
```

Si pasas este programa en tu ordenador, verás que en la pantalla se imprime:

```
1 1
1 2
1 3
2 1
2 2
2 3
```

Mientras la I del ciclo externo toma el valor 1, la J del ciclo interno recorre los valores 1, 2, 3; por tanto se imprime:

```
1  1
1  2
1  3
```

Cuando I toma el valor 2, J vuelve a valer nuevamente 1, 2 y 3, y se imprime:

```
2  1
2  2
2  3
```

A estos ciclos se les llama *encajados* o *anidados.*

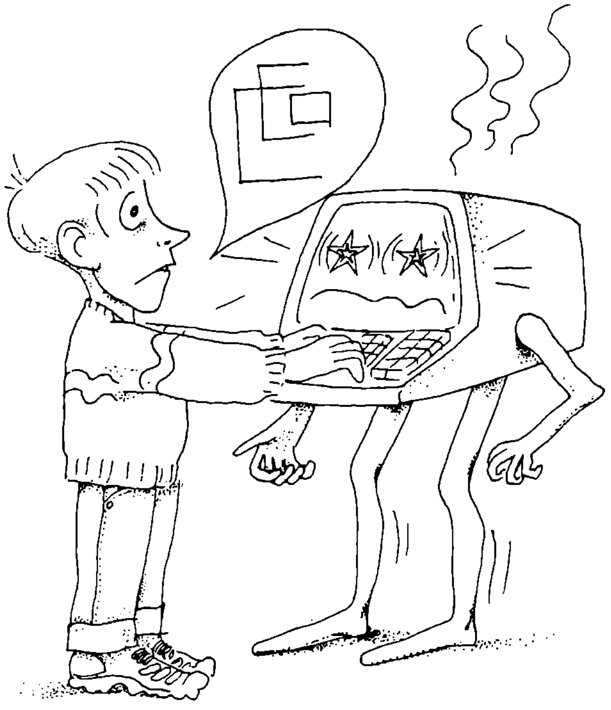
Ten cuidado, porque puedes cometer el siguiente error de construcción:

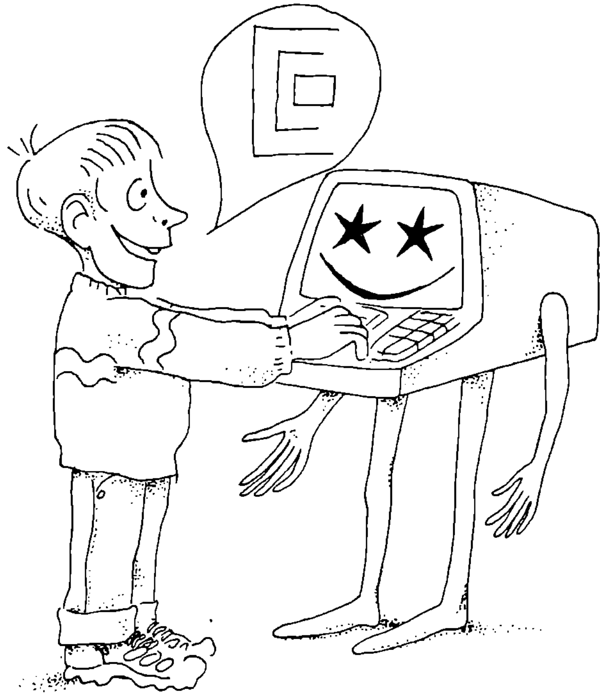
```
FOR I=1 TO 2
 FOR J=1 TO 3
 ......
 ......
 NEXT I
NEXT J
```

Recuerda que la construcción correcta es: El primer bucle que se abre es el último que se cierra, el segundo que se abre es el penúltimo que se cierra, y así sucesivamente.

Por tanto, el esquema de la construcción correcta es:

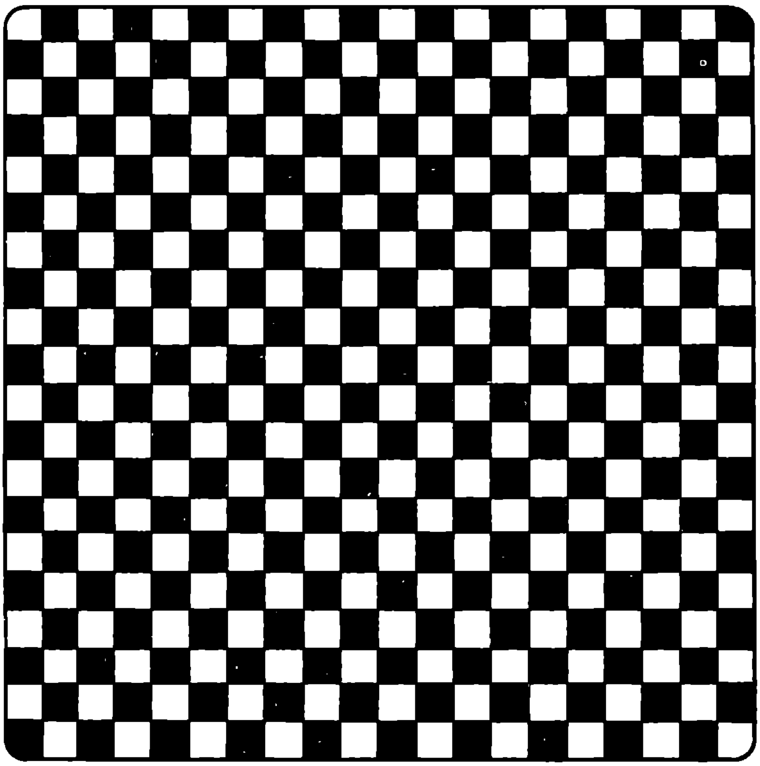
```
FOR I
 FOR J
 ......
 ......
 NEXT J
NEXT I
```





19.2 ALFOMBRA AJEDREZ

Como nos gusta mucho el ajedrez vamos a dibujar una alfombra ajedrez de veinte casillas por cada lado.



Podríamos hacerlo del mismo modo que el ajedrez, pero vamos a seguir un procedimiento distinto.

En la primera fila el bloque  se repite diez veces y en la segunda fila el bloque  se repite otras diez veces.

```
FOR J=1 TO 10
  PRINT "□■";
NEXT J
```

Con este ciclo dibujamos la primera fila.

```
FOR J=1 TO 10
  PRINT "■□";
NEXT J
```

Con este otro ciclo dibujamos la segunda fila.

Como hay que repetir estos dos ciclos diez veces, podemos incluirlos en un ciclo "madre".

```
10 REM ALFOMBRA
20 FOR I=1 TO 10
30 FOR J=1 TO 10
40  PRINT "□■";
50 NEXT J
60 PRINT
70 FOR J=1 TO 10
80  PRINT "■□";
90 NEXT J
100 PRINT
110 NEXT I
120 END
```

La instrucción PRINT de las líneas 60 y 100 hace que después de dibujada la fila correspondiente, el cursor salte a la línea siguiente; de lo contrario seguiría imprimiendo a continuación, y se estropearía el dibujo.

EJERCICIOS

19.1 Utiliza dos ciclos encajados para dibujar este rectángulo:

```
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
```

19.2 Con dos ciclos convenientemente encajados puedes representar este cuadro numérico:

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

19.3 Con dos ciclos encajados escribe dos tablas de multiplicar; por ejemplo, la del 7 y la del 8.

Modifica adecuadamente el programa anterior para escribir la del 7 y la del 9.

19.4 Como sabemos que te gusta la sopa, en las líneas DATA te damos los ingredientes para que cocines una buena sopa de letras.

```
200 DATA A,A,Z,E,R,E,C,J,I,B
210 DATA A,D,E,O,N,D,I,D,P,A
220 DATA L,A,N,O,L,E,M,U,R,A
230 DATA N,H,R,V,R,A,S,O,Z,A
240 DATA T,M,N,I,S,J,A,L,I
```

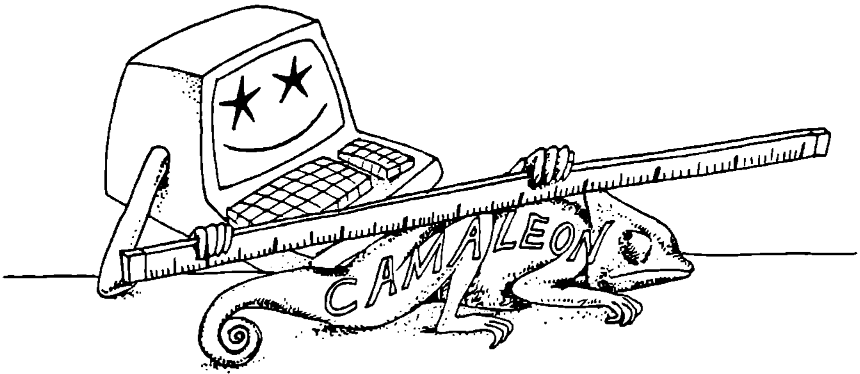
Con dos ciclos anidados lee y escribe las letras formando un cuadro de 7×7. En él has de encontrar seis frutas.

20

FUNCIONES PARA TRABAJAR CON PALABRAS

20.1 LA FUNCION LEN (LENGTH=LONGITUD)

¿Cuántas letras tiene la palabra COCODRILO? Las cuentas y ves que son 9 letras. El ordenador puede hacerlo exactamente igual que tú, por medio de la función LEN.



Escribe el siguiente programa:

```
10 REM CUENTO LETRAS
20 LET A$="COCODRILO"
30 LET N=LEN(A$)
40 PRINT N
50 END
```

Al ejecutarlo, aparece en pantalla:

```
RUN
9
READY.
■
```

Obtendrías el mismo resultado, si hubieses escrito el siguiente programa:

```
10 PRINT LEN("COCODRILO")
20 END
```

La función LEN no sólo cuenta letras, sino también los demás caracteres, por ejemplo, los espacios en blanco. Teclea el siguiente programa:

```
10 LET B$="YO SOY"
20 PRINT LEN(B$)
30 END
```

En la pantalla aparecerá:

```
RUN
6
READY.
■
```

Son cinco letras, y en la pantalla aparece un seis. Como ya dijimos antes, la función LEN cuenta todos los caracteres de la frase, y entre ellos está el espacio en blanco que hay entre YO y SOY. Es importante que lo recuerdes, porque vamos a estudiar más funciones para trabajar con cadenas y siempre debes considerar los espacios en blanco.

20.2 LA FUNCION LEFT$ (LEFT=IZQUIERDA)

La función LEFT$ toma de la cadena tantos caracteres, por la izquierda, como le indiquemos. Veamos un ejemplo:

```
10 REM COJO LETRAS POR LA IZQUIERDA
20 LET C$="CAMALEON"
30 PRINT LEFT$(C$,4)
40 END
```

En la pantalla, al ejecutarlo aparece:

```
RUN
CAMA
READY.
■
```

Es decir, de la palabra CAMALEON ha extraído las cuatro primeras letras empezando por la izquierda.

Antes de seguir debemos advertirte que hay ordenadores que no tratan así las cadenas; si ocurre esto en el tuyo, debes mirar el final de este capítulo donde se explica.

Pon en la cadena C$ la palabra "SALTAMONTES", y en la línea 30 un 5 en lugar de un 4. En la pantalla aparecerá ahora:

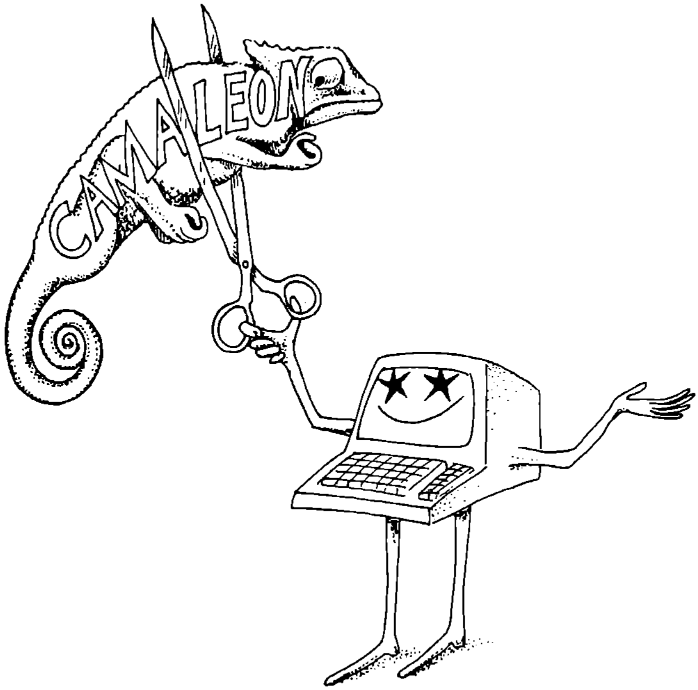
```
RUN
SALTA
READY.
■
```

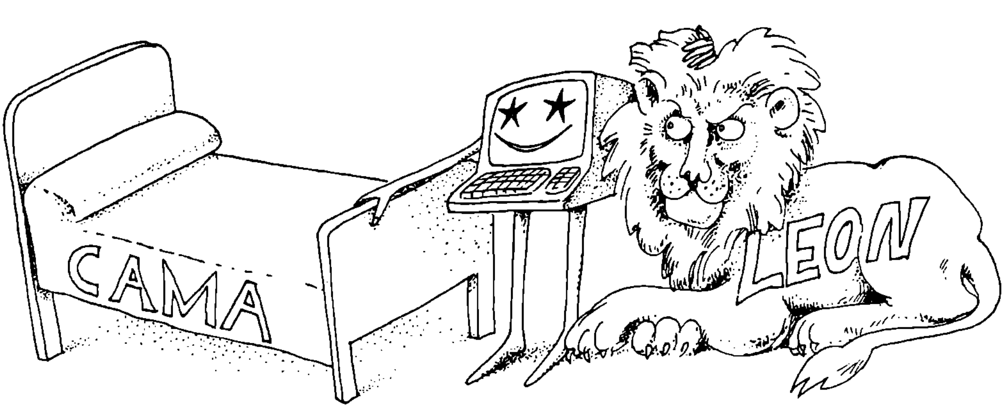
Con el siguiente programa te darás cuenta de que la función LEFT$ no sólo trabaja con palabras o frases, sino que también lo hace con cualquier cadena de caracteres o símbolos. Teclea el programa:

```
10 REM DIBUJO DE ASTERISCOS
20 LET A$="******"
30 FOR I=0 TO 5
40 PRINT LEFT$(A$,6-I)
50 NEXT I
60 END
```

En la pantalla tendrás:

```
RUN
******
*****
****
***
**
*
READY.
■
```


CAMALEON


CAMA
LEON

20.3 LA FUNCION RIGHT$ (RIGHT=DERECHA)

Esta función toma de la cadena tantos caracteres por la derecha como le indiquemos. Tenemos un ejemplo en el siguiente programa:

```
10 REM COJO LETRAS POR LA DERECHA
20 LET A$="CAMALEON"
30 PRINT RIGHT$(A$,4)
40 END
```

En la pantalla sale:

```
RUN
LEON
READY.
■
```

Es decir, aparece la palabra LEON, que está formada por las cuatro últimas letras de CAMALEON.

Veamos otro programa:

```
10 REM OTRO DIBUJO DE ASTERISCOS
20 LET A$="******"
30 FOR I=1 TO 6
40 PRINT RIGHT$(A$,I)
50 NEXT I
60 END
```

En la pantalla aparecerá así:

```
RUN
*
**
***
****
*****
******
READY.
■
```

20.4 LA FUNCION MID$ (MIDDLE=MEDIO)

Esta función aísla caracteres "en medio" de la palabra, pero es necesario decirle, no sólo cuántos ha de tomar, sino también dónde ha de empezar la cuenta. Por ejemplo:

```
10 LET D$="CAMALEON"
20 PRINT MID$(D$,2,3)
30 END
```

En la pantalla obtendrás:

```
RUN
AMA
READY.
■
```

Lo que nos indica que hemos empezado en el segundo carácter. que corresponde a la letra A, y hemos tomado tres caracteres.

20.5 EJEMPLOS SOBRE LAS FUNCIONES QUE TRABAJAN CON PALABRAS

Con los programas que te ofrecemos a continuación, pretendemos que manejes las funciones estudiadas en el párrafo anterior, así como otras cuestiones que has aprendido hasta ahora.

Escribamos un programa que presente en pantalla una palabra, de modo que aparezca en la primera línea una letra, en la siguiente dos, y así sucesivamente hasta completar la palabra.

```
10 REM APAREZCO POR LA IZQUIERDA
20 INPUT "PALABRA";P$
30 FOR I=1 TO LEN(P$)
40 PRINT LEFT$(P$,I)
50 NEXT I
60 END
```

Si ejecutas el programa y pones la palabra CANGURO, tienes en la pantalla:

```
RUN
PALABRA? CANGURO
C
CA
CAN
CANG
CANGU
CANGUR
CANGURO
READY.
■
```

Si utilizamos la función RIGHT$, vemos que la palabra aparece de derecha a izquierda. El programa que tienes que teclear es:

```
10 REM APAREZCO POR LA DERECHA
20 INPUT "PALABRAS";P$
30 FOR I=1 TO LEN(P$)
40 PRINT RIGHT$(P$,I)
50 NEXT I
60 END
```

En la pantalla saldrá:

```
RUN
PALABRA? CORBATA
A
TA
ATA
BATA
RBATA
ORBATA
CORBATA
READY.
■
```

Ahora escribe el siguiente programa:

```
10 REM PALABRA ESCRITA EN VERTICAL
20 INPUT "PALABRA";P$
30 FOR I=1 TO LEN(P$)
40 PRINT MID$(P$,I,1)
50 NEXT I
60 END
```

En la pantalla aparece:

```
RUN
PALABRA? CUERVO
C
U
E
R
V
O
READY.
■
```

20.6 LAS PALABRAS DEL SPECTRUM

Este ordenador posee una única función para el tratamiento de palabras; manejándola convenientemente puedes conseguir los mismos resultados que con las tres funciones anteriores. Así, por ejemplo, si A$="CAMIONES", tenemos:

```
A$(1 TO 6)=LEFT$(A$,6)="CAMION"
A$(4 TO 8)=RIGHT$(A$,5)="IONES"
A$(3 TO 4)=MID$(A$,3,2)="MI"
A$(5 TO 5)=MID$(A$,5,1)="O"
```

Con SALTAMONTES, puedes hacer el siguiente programa:

```
10 LET A$="SALTAMONTES"
20 PRINT A$(1 TO 5)
30 PRINT A$(6 TO 11)
40 PRINT A$(2 TO 4)
50 PRINT A$(8 TO 8)
60 END
```

Lo que nos dará en la pantalla:

```
RUN
SALTA
MONTES
ALT
N
READY.
■
```

EJERCICIOS

20.1 Prepara un programa que, mediante la función MID$, escriba la palabra PAPEL en vertical, tanto de arriba abajo, como de abajo hacia arriba.

20.2 Prepara un programa que, mediante la función MID$, escriba la palabra MARGARITA al revés.

20.3 Prepara un programa que escriba el plural de las palabras terminadas en IZ, como lápiz, perdiz, o nariz.

20.4 Diseña un programa para que el ordenador pregunte el nombre y los dos apellidos de una persona; y como respuesta escriba la inicial del nombre y los apellidos completos. (Si le das RAFAEL ALONSO PEREZ, escribirá R. ALONSO PEREZ).

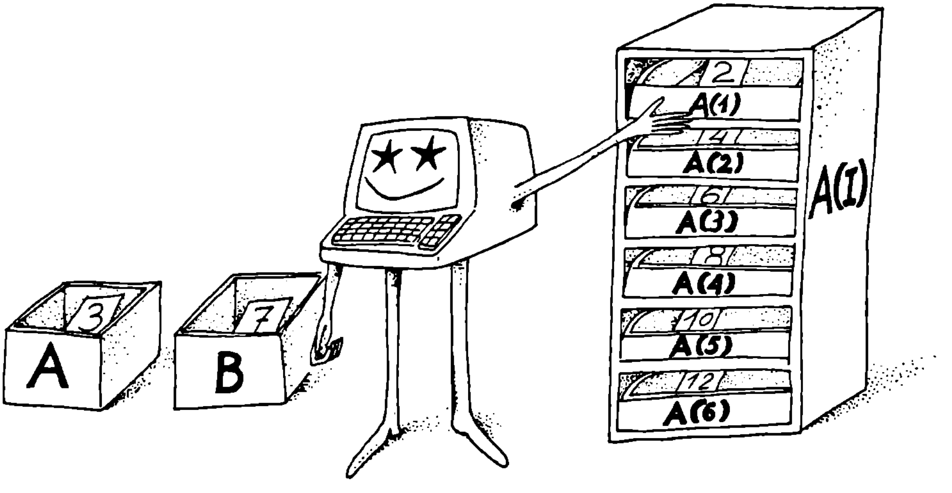
21

LISTAS

21.1 VARIABLES DE INDICE

Ya sabes que una variable es como una caja en la que se puede guardar un número. Así, por ejemplo, las instrucciones LET A=3 y LET B=7 crean las cajas A y B, y guardan en ellas los números 3 y 7.

Ahora vamos a conocer una posibilidad muy interesante: la de poder unir varias cajas para formar cajoneras:



Observa que todos los cajones de la cajonera llevan un nombre parecido: A(1), A(2),..., A(6).

Cada una de estas cajoneras es una *variable de índice;* el conjunto de todas ellas forma una *lista*. Cualquiera de las cajas puede ser representada por A(I). Cuando I es 1, entonces A(I) representa a A(1); cuando I es 2, A(I) representa a A(2), y así sucesivamente.

Observa que hay variables que tienen un nombre parecido, como A1 y A(1), y son distintas. La primera es una caja "aislada", la segunda es un cajón de la cajonera A(I). Las variables AI y A(I) también son distintas. El *paréntesis* es lo que distingue a las variables de índice de las que no lo son.

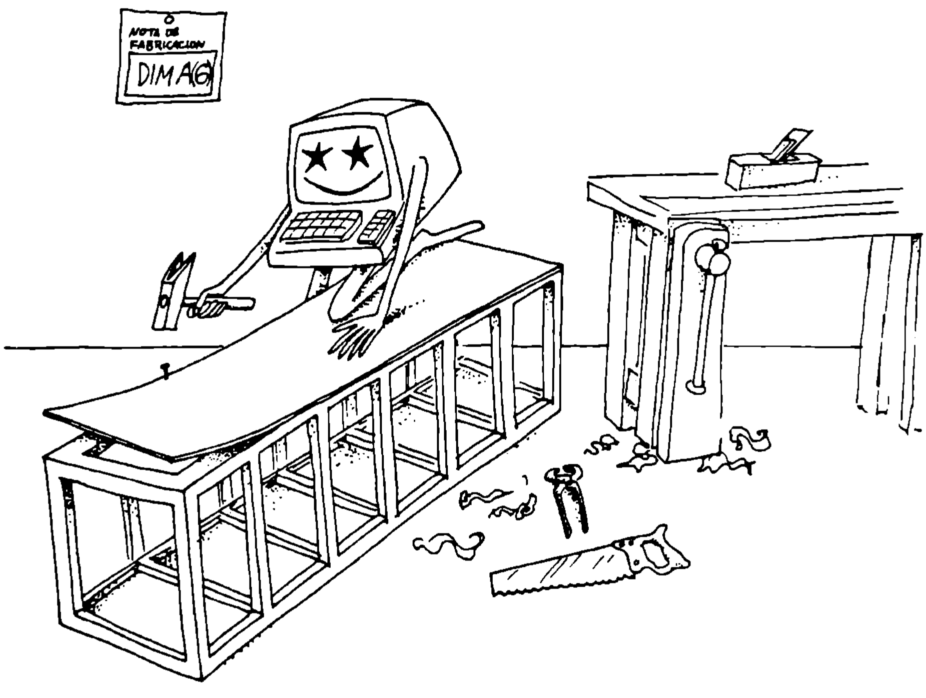
El nombre de una variable de índice está formado por el nombre de una variable más un número entre paréntesis:

A(3)   B(7)   A1(3)   AB(7)   LADO(5)

son nombres de variables de índice.

21.2 INSTRUCCION DIM

Para que el ordenador nos prepare una cajonera y podamos utilizarla, es necesario advertírselo en las primeras líneas del programa.



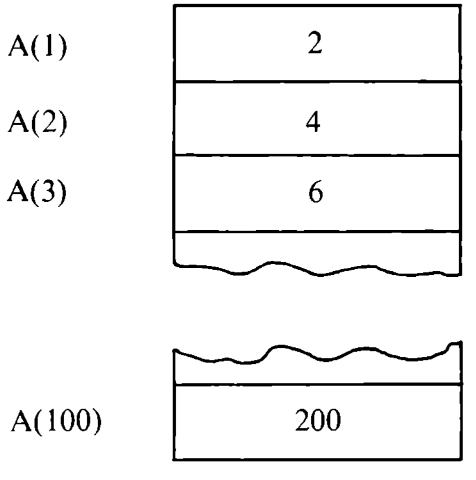
Esto se hace con la instrucción DIM (dimensión de la cajonera o de la lista). Así, la línea de programa

20 DIM A(100)

hace que el ordenador nos prepare una lista con cien variables A(1), A(2),..., A(100).

21.3 UN EJEMPLO

El hecho de que en A(I) el índice I sea variable, da a las variables de índice toda su potencia. Veamos como ejemplo, lo cómodo que resulta asignar los valores 2, 4, 6, ..., 200 a una lista de cien variables:



Observa que en cada cajón hay que meter un número que es el doble del índice. Es decir, en el cajón I hay que poner 2 * I.

El programa puede ser éste:

```
10 REM CARGAR UNA LISTA
20 DIM A(100)
30 FOR I=1 TO 100
40 LET A(I)=2*I
50 NEXT I
```

Te puede quedar la sospecha de si la lista ha sido correctamente cargada o no. Para salir de dudas vamos a pedir al ordenador que nos la escriba, para lo cual prolongamos el programa con la líneas siguientes:

```
60 REM ESCRITURA DE LA LISTA
70 FOR I=1 TO 100
80 PRINT A(I);
90 NEXT I
100 END
```

Comprueba que en la pantalla aparece la serie 2, 4, 6, ..., 200.

21.4 RECUENTO DE LOS PUNTOS OBTENIDOS AL LANZAR UN DADO MIL VECES

A continuación vas a tener una prueba de cómo el empleo de variables de índice simplifica y reduce los programas.

En el apartado 18.2 hicimos un programa que nos decía cuántas veces había aparecido cada cara de un dado, después de haberlo lanzado mil veces. Veamos ahora cómo se puede mejorar ese programa.

Suponiendo que el resultado obtenido en cada tirada sea D, el contador:

LET C(D)=C(D)+1

vale por seis contadores particulares: C(1) cuenta los unos, C(2) cuenta los doses, ..., y C(6) cuenta los seises.

El programa de la página anterior puede quedar, ahora, así:

```
10 REM RECUENTO DE LOS PUNTOS OBTENIDOS
20 REM AL LANZAR UN DADO MIL VECES
30 REM INICIALIZACION DE CONTADORES
40 DIM C(6)
50 FOR D=1 TO 6:LET C(D)=0:NEXT D
60 REM MIL LANZAMIENTOS
70 FOR I=1 TO 1000
80 LET D=INT(6*RND(1))+1
```

```
90 LET C(D)=C(D)+1
100 NEXT I
110 REM ESCRITURA DEL RECUENTO
120 FOR D=1 TO 6
130 PRINT "EL";D;"HA SALIDO";C(D); "VECES"
140 NEXT D
150 END
```

21.5 VARIABLES DE CADENA CON INDICE

También existen variables de índice en las que se pueden guardar cadenas de caracteres, como por ejemplo el nombre de una persona.

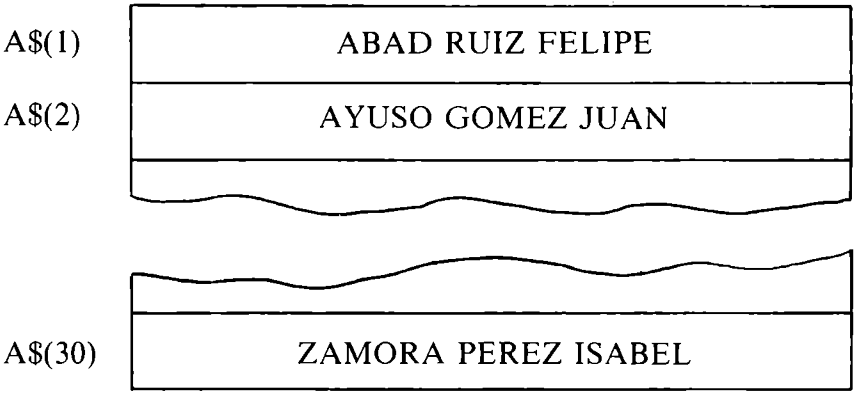
Para formar el nombre de una *variable de cadena con índice* tienes que incluir el signo del dólar.

La serie de variables:

A$(1), A$(2), ..., A$(30)

constituye una lista de dimensión 30, en la que puedes guardar el nombre y los apellidos de tus compañeros de clase:

LISTA A$(I)



¿Cómo puedes cargar esta lista en tu ordenador? El siguiente programa sirve para introducir, uno a uno, los treinta nombres de la lista:

```
10 REM CARGAR UNA LISTA DE NOMBRES
20 DIM A$(30)
30 FOR I=1 TO 30
40 INPUT "NOMBRE: ";A$(I)
50 NEXT I
60 END
```

La ejecución del programa se detendrá en espera de que escribas el primer nombre, luego se volverá a detener para que escribas el segundo nombre, y así, hasta el último.

Haz un par de ensayos para comprobar que ha cargado la lista. Escribe:

PRINT A$(1) [RETURN]
PRINT A$(30) [RETURN]

Tras estas órdenes, inmediatamente aparecerán en la pantalla los nombres del primero y del último de la lista.

EJERCICIOS

21.1 Prepara un programa que cargue los nombres de tus compañeros de clase y que luego escriba la lista completa.

21.2 En la lista M(I) están las notas de matemáticas de once alumnos, y en la L(J) las notas de lengua. Haz un programa que calcule la media de las notas de matemáticas y la de las notas de lengua.

21.3 Mete en una lista los nombres de doce amigos. Haz un programa para que el ordenador seleccione los nombres de cinco de ellos para formar un equipo de baloncesto.

21.4 Las disposiciones electorales españolas prevén la aplicación de la ley d'Hont a la hora de distribuir los escaños entre los partidos, según los votos que se hayan obtenido.

El motivo de hacerlo así es evitar la dispersión del electorado en muchos partidos pequeños.

El siguiente programa, después de preguntar los datos necesarios, calcula los escaños que corresponden a cada partido en unas elecciones.

Ejecútalo varias veces. Puedes mejorarlo para que sea más rápido.

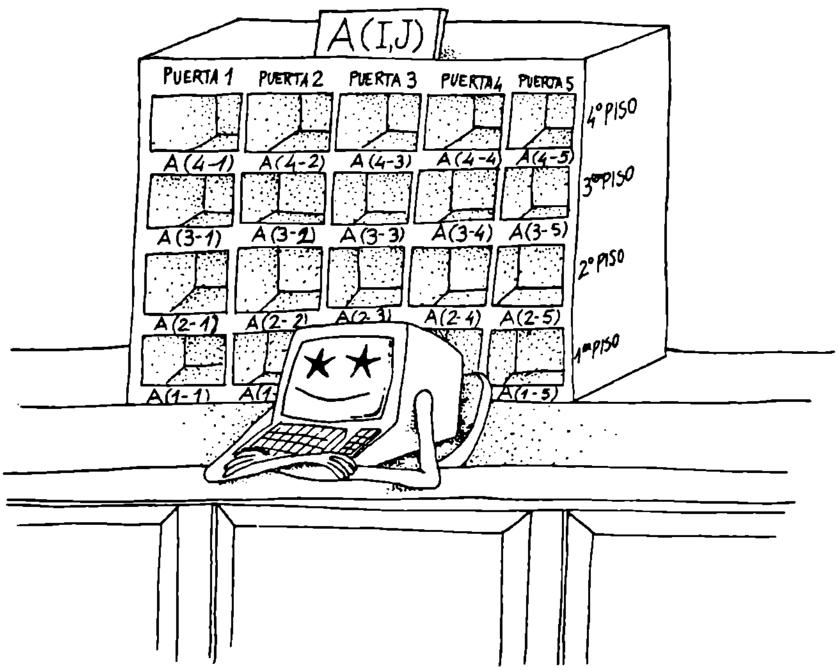
```
10 REM ELECCIONES
20 INPUT "NUMERO DE PARTIDOS";P
30 INPUT "ESCAÑOS A DISTRIBUIR";E
40 DIM P$(P),V(P),A(P),E(P)
50 REM ENTRADA DE DATOS
60 FOR I=1 TO P
70 INPUT "NOMBRE DEL PARTIDO";P$(I)
80 INPUT "VOTOS OBTENIDOS";V(I)
90 NEXT I
100 REM REPARTO DE LOS ESCAÑOS
110 FOR X=1 TO E
120 REM VARIABLE AUXILIAR
130 FOR I=1 TO P
140 LET A(I)=V(I)/(E(I)+1)
150 NEXT I
160 REM CALCULO DEL MAXIMO
170 LET M=A(I)
180 FOR I=2 TO P
190 IF A(I)>M THEN M=A(I)
200 NEXT I
210 REM EL MAXIMO ES M
220 REM SE LE AÑADE UN ESCAÑO
230 E(M)=E(M)+1
240 NEXT X
250 REM PRESENTACION DE RESULTADOS
260 FOR I=1 TO P
270 PRINT P$(I),V(I),E(I)
280 NEXT I
290 END
```

22

TABLAS

22.1 CASILLEROS

El lenguaje BASIC permite utilizar *variables con dos índices*, tanto numéricas como de cadenas de caracteres. Puedes imaginarte que el conjunto de estas variables forma un casillero como el de la recepción de un hotel:



El casillero está organizado por filas o pisos: 1º, 2º, 3º y 4º piso; y dentro de cada piso las casillas están numeradas del 1 al 5.

En adelante representaremos el casillero por una tabla de dos dimensiones (en nuestro ejemplo, de 4 filas y 5 columnas) en la que, como es costumbre establecida, numeraremos las filas de arriba a abajo:

	1ª columna	*2ª columna*	*3ª columna*	*4ª columna*	*5ª columna*
1ª fila	A(1,1)	A(1,2)	A(1,3)	A(1,4)	A(1,5)
2ª fila	A(2,1)	A(2,2)	A(2,3)	A(2,4)	A(2,5)
3ª fila	A(3,1)	A(3,2)	A(3,3)	A(3,4)	A(3,5)
4ª fila	A(4,1)	A(4,2)	A(4,3)	A(4,4)	A(4,5)

Cada una de las variables de la tecla lleva dos índices: el primero indica la fila a la que pertenece y el segundo la columna. Así, por ejemplo, debes imaginar la variable A(2,4) situada en la segunda fila y cuarta columna.

Como índices se suelen utilizar las letras I y J. Así, A(I,J) es la variable A(2,4) cuando I=2 y J=4.

Tendrás que hacer ahora lo mismo que hacías cuando trabajabas con variables de un solo índice: advertir al ordenador del tamaño de la tabla que vas a usar. Esto se hace con la instrucción DIM. Por ejemplo, la línea:

```
20 DIM A(4,5)
```

hace que el ordenador te prepare una tabla de 4 filas y 5 columnas.

22.2 LAS TABLAS DE MULTIPLICAR

Veamos cómo se puede cargar en el ordenador la siguiente tabla:

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

Esta tabla agrupa las nueve tablas de multiplicar. En cada posición figura el producto de los dos índices:

$$A(I,J) = I * J$$

El siguiente programa va a cargar la tabla por filas:

```
10 REM TABLA
20 DIM A(9,9)
30 FOR I=1 TO 9
40 FOR J=1 TO 9
50 LET A(I,J)=I*J
60 NEXT J
70 NEXT I
```

Observa cómo funcionan los dos ciclos anidados FOR-NEXT: Para I=1, J tomará los valores 1, 2, ..., 9 y se cargará la primera fila de la tabla; para I=2, J volverá a valer 1, 2, ..., 9 y entrará la segunda fila, y así sucesivamente, hasta llegar a I=9, que es cuando se carga la última fila.

Si ahora quieres que aparezca la tabla en la pantalla, puedes completar el programa con las siguientes líneas:

```
80 REM ESCRITURA DE LA TABLA
90 FOR I=1 TO 9
100 FOR J=1 TO 9
110 PRINT TAB(4*J);A(I,J);
120 NEXT J
130 PRINT
140 NEXT I
150 END
```

La instrucción PRINT de la línea 130 se necesita para romper el efecto del punto y coma del final de la línea 110, y poder pasar de una fila a la siguiente. (Suprime la línea 130 y observa cómo queda la tabla).

Puede ocurrir que la pantalla de tu ordenador no sea lo suficientemente ancha como para que quepa toda la tabla. Si es así, modifica el programa; por ejemplo, puedes sustituir la línea 100 por esta otra:

```
100 FOR J=1 TO 5
```

En este caso aparecerán únicamente las cinco primeras columnas.

Los microordenadores también permiten el manejo de tablas cuyos

elementos sean cadenas de caracteres. Para ello deberás incluir en el nombre de la variable el símbolo del dólar: A$(I,J).

EJERCICIOS

22.1 El cuadro numérico

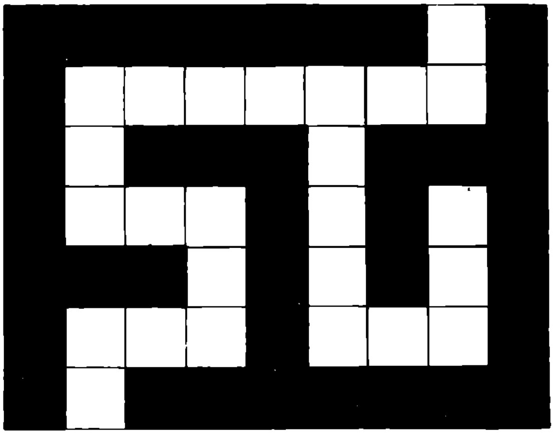
```
1 1 1 1 1 1 1 0 1
1 0 0 0 0 0 0 0 1
1 0 1 1 1 0 1 1 1
1 0 0 0 1 0 1 0 1
1 1 1 0 1 0 1 0 1
1 0 0 0 1 0 0 0 1
1 0 1 1 1 1 1 1 1
```

se carga en la tabla T(I,J) con el siguiente programa:

```
10 REM CUADRO NUMERICO
20 DIM T(7,9)
30 FOR I=1 TO 7
40 FOR J=1 TO 9
50 READ T(I,J)
60 NEXT J
70 NEXT I
80 DATA 1,1,1,1,1,1,1,0,1
90 DATA 1,0,0,0,0,0,0,0,1
100 DATA 1,0,1,1,1,0,1,1,1
110 DATA 1,0,0,0,1,0,1,0,1
120 DATA 1,1,1,0,1,0,1,0,1
130 DATA 1,0,0,0,1,0,0,0,1
140 DATA 1,0,1,1,1,1,1,1,1
```

Añade otro ciclo anidado parecido al anterior que imprima en la pantalla el cuadro numérico. (Entre las líneas NEXT J y NEXT I necesitarás poner una instrucción PRINT como en el programa ESCRITURA DE LA TABLA del párrafo 22.2.)

22.2 El cuadrado numérico del ejercicio anterior es el código secreto de este laberinto:



en el que el 1 significa "pared" y el 0 "paso libre". Modifica el programa del ejercicio anterior para que, en lugar de escribir el cuadrado numérico, el ordenador dibuje el laberinto.

Cambia algún número en las líneas DATA y observa el efecto producido en el dibujo.

22.3 En un hotel de cinco plantas las habitaciones tienen un número de dos cifras, la primera corresponde a la planta y la segunda a su situación en el pasillo. Un corte esquemático del edificio es éste:

51	52	53	54	55	56	57
41	42	43	44	45	46	47
31	32	33	34	35	36	37
21	22	23	24	25	26	27
11	12	13	14	15	16	17

RECEPCION

Haz un programa que lo represente en la pantalla del ordenador.

22.4 El ordenador de la recepción del hotel emplea una tabla para saber cuáles de las habitaciones están libres y cuáles ocupadas. Prepara un programa que ponga un cero a las habitaciones libres y un uno a las ocupadas, permitiendo reservar así plazas o dejar habitaciones libres, sin más que indicar el piso y la puerta.

23

SUBRUTINAS

23.1 SUBRUTINAS

En ocasiones tenemos que repetir muchas veces una misma tarea, ¡vaya cosa más pesada! Lo mismo le puede pasar al ordenador, y lo peor es que como somos nosotros los que le decimos qué tiene que hacer, tendremos que darle muchas veces las mismas instrucciones. Para aliviar un poco esta situación, se puede separar la tarea que se va a repetir y para usarla "se envía un mensaje a la dirección donde reside".

A estas tareas, a las que podemos llamar desde cualquier parte del programa, les llamamos SUBRUTINAS.

23.2 SAN SERENIN DEL MONTE

Vamos a emplear una subrutina para ahorrarnos trabajo con una vieja canción infantil, "San Serenín del Monte". Es así:

> San Serenín del Monte,
> San Serenín, cortés;
> yo, como buen cristiano,

yo, me levantaré.

> San Serenín del Monte,
> San Serenín, cortés;
> yo, como buen cristiano,

yo, me persignaré.

San Serenín del Monte,
San Serenín, cortés;
yo, como buen cristiano,

yo, te saludaré.

Como ves tiene doce versos, repartidos en tres estrofas, de manera que los tres primeros versos de cada estrofa se repiten, y sólo varía el cuarto. Podemos usar una subrutina que escriba los tres primeros versos, y ésos sólo tendríamos que escribirlos una vez y no tres. El diagrama de flujo será:

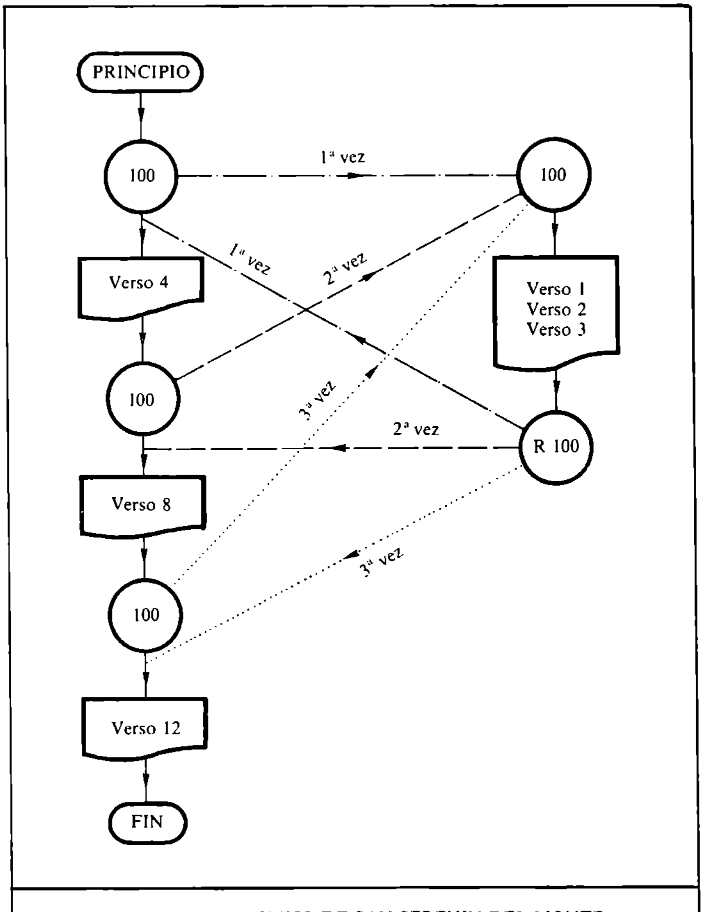
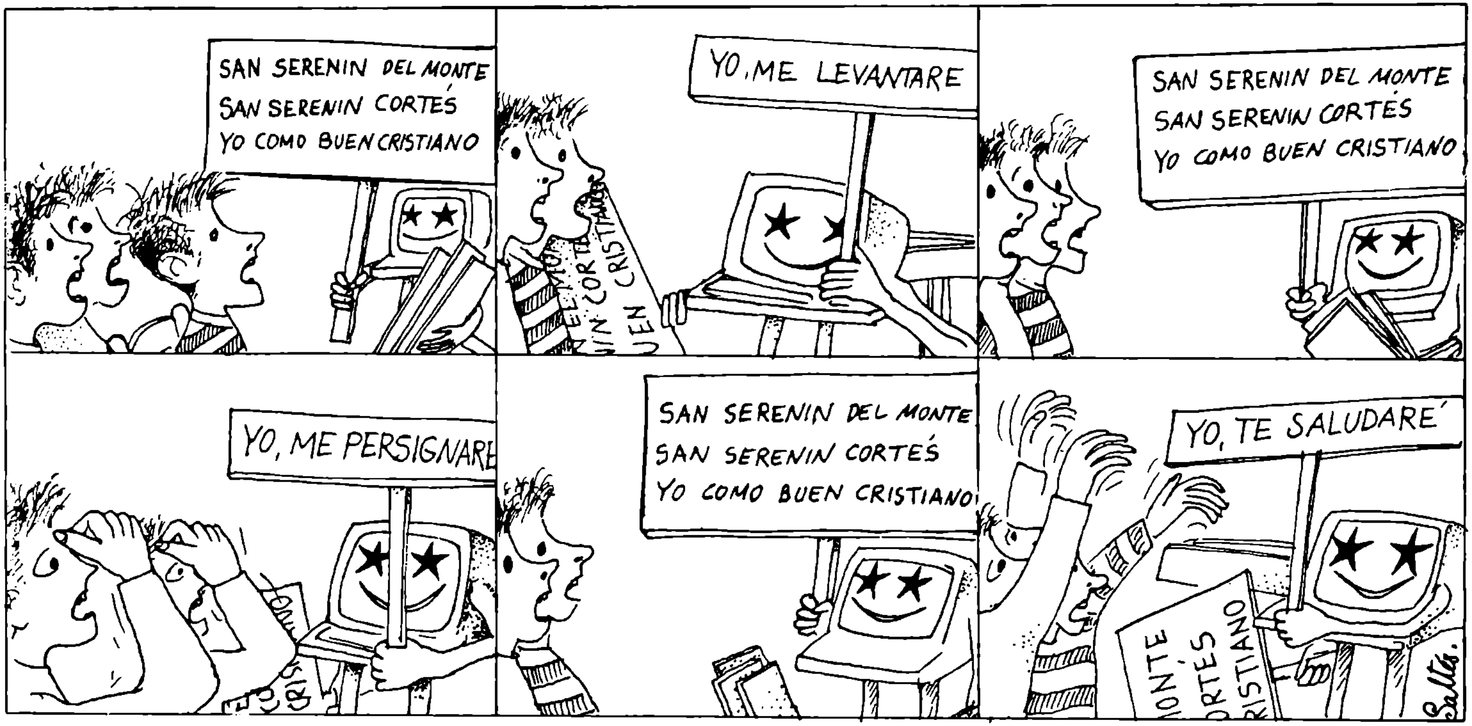


DIAGRAMA DE FLUJO DE SAN SERENIN DEL MONTE


SAN SERENIN DEL MONTE
SAN SERENIN CORTÉS
YO COMO BUEN CRISTIANO
YO, ME LEVANTARE
SAN SERENIN DEL MONTE
SAN SERENIN CORTÉS
YO COMO BUEN CRISTIANO
YO, ME PERSIGNARE
SAN SERENIN DEL MONTE
SAN SERENIN CORTÉS
YO COMO BUEN CRISTIANO
YO, TE SALUDARÉ
Saltis.

Como indican los círculos del diagrama principal, hay tres lugares desde los cuales "se llama a la subrutina". Al llegar a ellos el ordenador salta a la subrutina memorizando dónde estaba y, al acabar de realizarla, regresa al lugar desde el cual había emprendido el viaje. El programa va a ayudarnos a aclarar esto:

```
10 REM SAN SERENIN DEL MONTE
20 GOSUB 100
30 PRINT "YO ME LEVANTARE."
40 GOSUB 100
50 PRINT "YO ME PERSIGNARE."
60 GOSUB 100
70 PRINT "YO TE SALUDARE."
80 END
100 REM SUBRUTINA
110 PRINT "SAN SERENIN DEL MONTE,"
120 PRINT "SAN SERENIN, CORTES;"
130 PRINT "YO, COMO BUEN CRISTIANO,"
140 RETURN
```

En la línea 20 el ordenador se ve forzado a pasar a la 100, y a recordar que antes de la 100 estaba en la 20; al llegar a la línea 100 se comporta de la forma habitual, va haciendo lo que se le indica en cada línea y pasando a la siguiente, hasta llegar a la 140, donde recibe orden de regresar a la línea siguiente a la 20, que es la 30. Igual se comporta las restantes veces.

23.3 LAS VARIABLES EN LAS SUBRUTINAS

En el programa anterior todo es constante: no hay ninguna variable cuyo valor se altere durante la ejecución del programa. Ahora vamos a ver un programa con dos variables, una cuyos valores se alteran sólo en el programa principal, y otra que sólo se altera en la subrutina. Crearemos un reloj digital que cuente segundos, que meteremos en la variable S, y décimas que cargaremos en la variable D. Los segundos variarán en el programa principal y las décimas en una subrutina, de acuerdo con el siguiente diagrama de flujo:

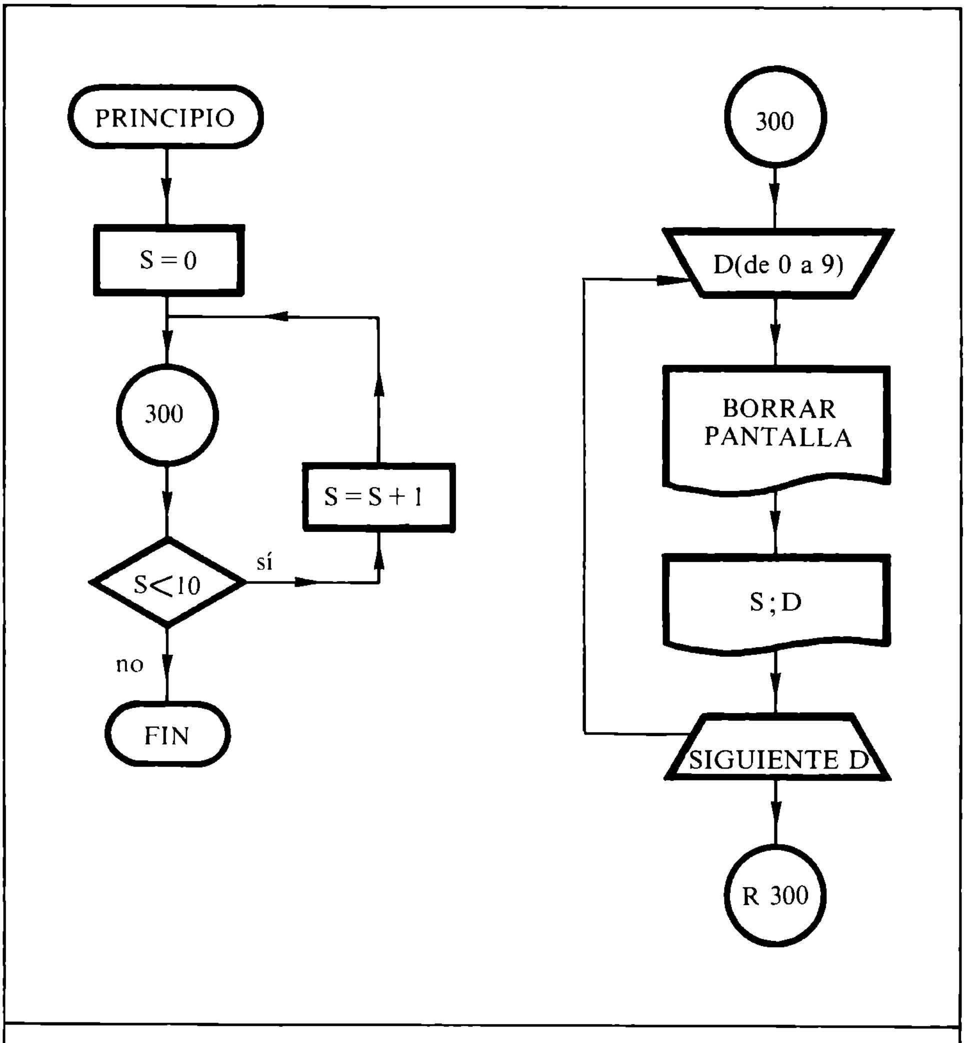


DIAGRAMA DE FLUJO DE RELOJ DIGITAL

El programa será éste:

```
10 REM RELOJ DIGITAL
20 LET S=0:REM SEGUNDOS
30 GOSUB 300:REM DECIMAS
40 IF S<10 THEN LET S=S+1:GOTO 30
50 END
300 REM SUBRUTINA DECIMAS
310 FOR D=0 TO 9
320 (BORRAR LA PANTALLA)
330 PRINT S;D
340 NEXT D:REM SIGUIENTE DECIMA
350 RETURN
```

EJERCICIOS

23.1 Hay una cancioncilla de Lope de Vega que dice así:

La Virgen de la Cabeza,
¡Quién como ella!
hizo gloria aquesta tierra.
¡Quién como ella!
Tiene la frente de perlas
¡Quién como ella!
y de oro fino las hebras.
¡Quién como ella!

Utiliza una subrutina para programar la escritura de estos versos.

23.2 El programa que sigue tiene por objeto tomar dos números y escribir la suma de ellos, la suma de sus cuadrados y la suma de sus cubos. Nosotros damos las líneas del programa; tú tienes que numerarlas, poner la dirección en las instrucciones GOSUB y, por último, probar el programa.

```
REM SUMAS
REM PROGRAMA PRINCIPAL
INPUT "DAME DOS NUMEROS";X,Y
LET A=X
LET B=Y
GOSUB
LET A=X*X
LET B=Y*Y
GOSUB
LET A=X*X*X
LET B=Y*Y*Y
GOSUB
END
```

```
REM SUBRUT.PARA SUMAR
LET S=A+B
PRINT "LA SUMA ES";S
RETURN
```

23.3 La programación modular es una técnica que consiste en descomponer un problema en varias partes llamadas *módulos,* programar por separado cada una de ellas, y por último, unirlas. Cada módulo se puede programar como una subrutina que es reclamada desde el programa principal, cuya misión es unir los distintos módulos.

En la programación de un juego puedes distinguir tres partes:

1ª PRESENTACION DEL JUEGO, que tiene por objeto explicar al usuario las reglas del juego.

2ª JUEGO propiamente dicho.

3ª FINAL DEL JUEGO, en donde el ordenador pregunta si el jugador quiere volver a jugar, en cuyo caso comienza otra vez el juego, o si quiere terminar.

El juego que tienes que programar con esta técnica es el siguiente:

a) Tirar dos dados.
b) Si la suma es 7 u 11, gana el ordenador; si no, ganas tú.

Te damos el programa principal. Tú tienes que escribir las tres subrutinas

```
10 REM DADOS
20 REM PROGRAMA PRINCIPAL
30 GOSUB 100
40 GOSUB 200
50 GOSUB 300
60 END
```

```
100 REM SUBRUTINA DE PRESENTACION
.................................
199 RETURN
```

```
200 REM JUEGOS
.................................
299 RETURN
```

```
300 REM FINAL DEL JUEGO
.................................
399 RETURN
```

23.4 Aplica las técnicas de la programación modular para mejorar el programa ADIVINA del capítulo 12.

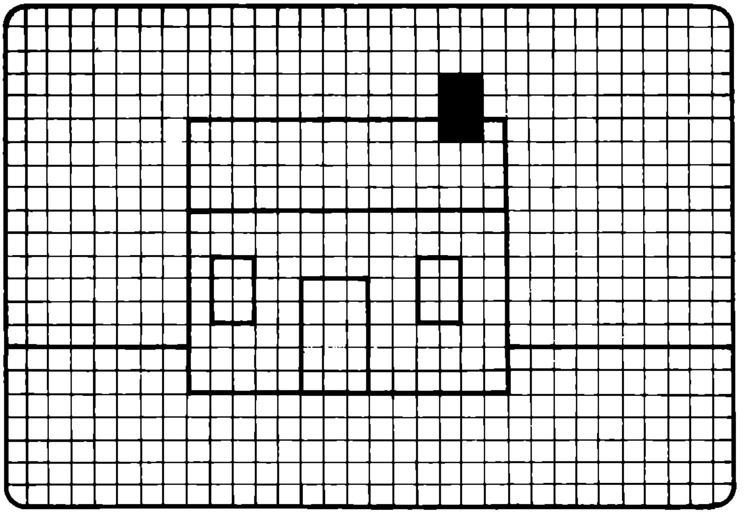
24

COMO DIBUJAR UNA CASA EN COLOR

24.1 PLANIFICACION

Las subrutinas son muy útiles para ahorrarse trabajo, para programar una sola vez aquellas cosas que se repiten. Pero no es necesario que algo se repita para que se pueda programar como subrutina. Muchas veces interesa, para mantener la claridad del programa, que algunas de sus partes se programen separadas como subrutinas en otro lugar del que les corresponde, y a continuación se ensamblen. Vamos a aclarar esto con un ejemplo que consistirá en pintar una casa de colores.

Para pintar en colores esta casa tenemos que hacer lo siguiente:



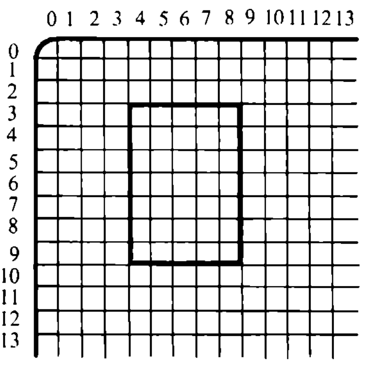
1. Poner la pantalla azul.
2. Poner el suelo verde.
3. Pintar la pared amarilla.
4. Poner el techo rojo.
5. Poner la puerta negra.
6. Poner las ventanas blancas.
7. Pintar la chimenea morada.

De estas siete cosas, seis consisten en pintar un rectángulo de algún color. Lo más cómodo va a ser enseñar al ordenador a pintar rectángulos y pedirle que los haga de distintos colores y tamaños, y en diferentes sitios. La cosa quedará así:

1. Poner la pantalla azul (Cielo).
2. Hacer abajo un rectángulo verde, de todo el ancho de la pantalla (Suelo).
3. Pintar otro rectángulo, amarillo, un poco más arriba y más estrecho (Pared).
4. Justo encima del anterior poner otro, rojo, de igual anchura (Tejado).
5. En el centro del amarillo, y llegando al extremo inferior, poner otro negro (Puerta).
6. En las zonas más despejadas del rectángulo amarillo, pintar otros dos blancos (Ventanas).
7. En lo alto del rectángulo rojo, poner uno morado (Chimenea).

24.2 LA CASA CON PRINT AT

Supón que tenemos que pintar, con el color correspondiente al número 2 (K = 2), un rectángulo de cinco casillas de ancho y siete de alto (AN = 5, AL = 7) y cuyo vértice superior izquierdo está en la casilla de la tercera fila y cuarta columna (FI = 3, CO = 4).



Si tu ordenador dispone de la instrucción PRINT AT, puedes emplear una subrutina con dos ciclos encajados, de manera que el exterior regule la fila en que se pinta y el interior la columna. Esto es lo que muestra el siguiente diagrama del flujo:

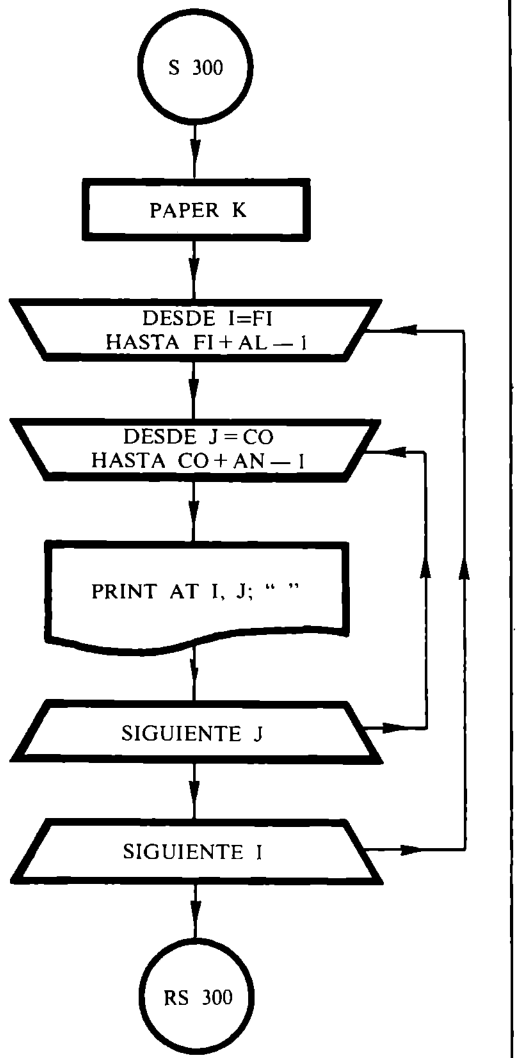


Diagrama del flujo de rutina
que PINTA RECTANGULOS

El programa correpondiente es éste:

```
300 REM PINTAR RECTANGULO CON INSTRUCCION "PRINT
    AT"
310 PAPER K
320 REM PARA CADA I SE PINTA UNA FILA
330 FOR I=FI TO FI+AL-1
340 REM CON CADA J SE PINTA UN CUADRADO DE LA FILA
350 FOR J=CO TO CO+AN-1
360 PRINT AT I,J;" "
370 NEXT J
380 NEXT I
390 RETURN
```

24.3 LA CASA SIN PRINT AT

Si tu ordenador carece de la instrucción mencionada, debes emplear el programa que sigue para crear el rectángulo de esas dimensiones y color. Como puedes apreciar es una ligera variante del anterior:

```
400 REM PINTAR RECTANGULO SIN INSTRUCCION "PRINT
    AT"
410 REM PARA CADA I SE PINTA UNA FILA
420 FOR I=FI TO FI+AL-1
430 REM CON CADA J SE PINTA UN CUADRADO DE LA FI
    LA
440 FOR J=CO TO CO+AN-1
450 PRINT TAB(J);K$;" ";
460 NEXT J
470 PRINT:REM BAJAR A OTRA FILA
480 NEXT I
490 RETURN
```

El único problema con este programa es que pinta el rectángulo pegado al borde superior de la pantalla. Para evitar esto, debes emplear previamente una subrutina que baje el cursor, sin escribir nada, FI veces. Esta subrutina responde al siguiente diagrama de flujo:

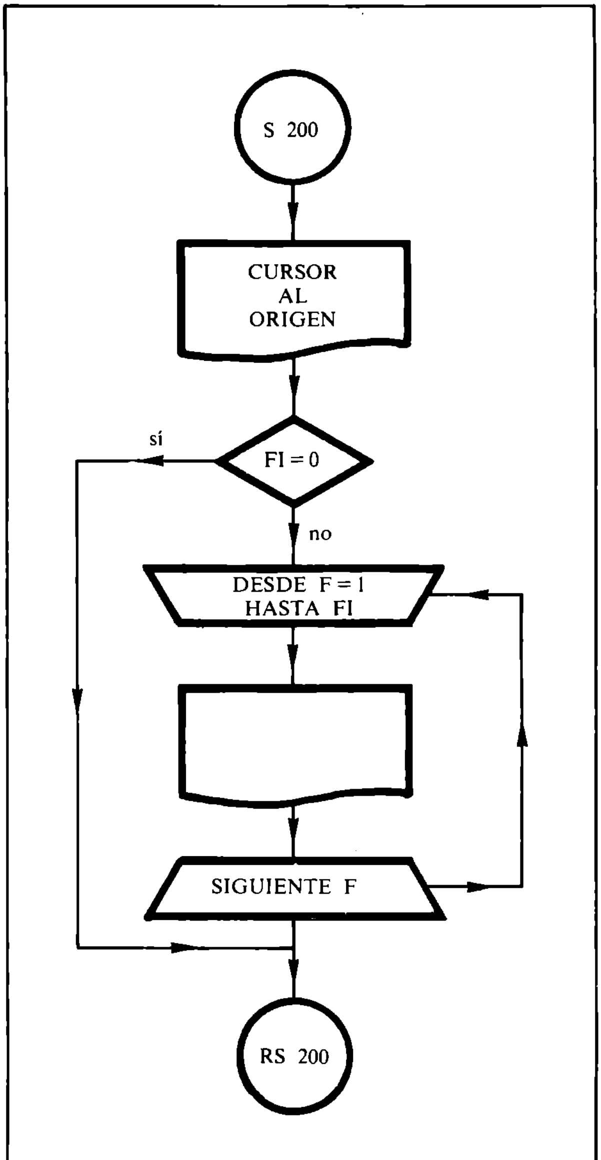


Diagrama de flujo de rutina
que BAJA A LA FILA NECESARIA

Esta es la lista de instrucciones que corresponde al diagrama de flujo:

```
200 REM BAJAR A LA PRIMERA FILA DEL RECTANGULO
210 PRINT "[HOME]"
220 REM SI LA PRIMERA FILA ES LA CERO, NO HAY QU
    E BAJAR
230 IF FI=0 THEN GOTO 270
240 FOR F=1 TO FI
250 PRINT
260 NEXT F
270 RETURN
```

24.4 PROGRAMA DEFINITIVO

El programa para el Spectrum, que dispone de la instrucción PRINT AT, es el siguiente:

```
10 REM CASA
20 REM BORRAR LA PANTALLA
30 PAPER 1:CLS
40 FOR R=1 TO 7
50 REM LEER LOS DATOS DEL RECTANGULO
60 READ K,AN,AL,FI,CO
70 GOSUB 300:REM PINTAR RECTANGULO
80 NEXT R
90 END
```

....
aquí debes poner la subrutina 300.
....

```
610 DATA 5,32,7,15,0:REM SUELO
620 DATA 6,14,8,9,8:REM PARED
630 DATA 2,14,4,5,8:REM TECHO
640 DATA 0,3,5,12,13:REM PUERTA
650 DATA 7,2,3,11,9:REM VENTANA 1
660 DATA 7,2,3,11,18:REM VENTANA 2
670 DATA 3,2,3,3,19:REM CHIMENEA
```

Si tu ordenador es un Commodore 64, el programa que has de usar es este otro:

```
10 REM CASA
20 REM BORRAR LA PANTALLA
30 PRINT "[SHIFT][HOME]"
40 FOR R=1 TO 7
```

```
50 REM LEER LOS DATOS DEL RECTANGULO
60 READ K$,AN,AL,FI,CO
70 GOSUB 200:REM BAJAR
80 GOSUB 400:REM PINTAR RECTANGULO
90 NEXT R
100 END
```

....

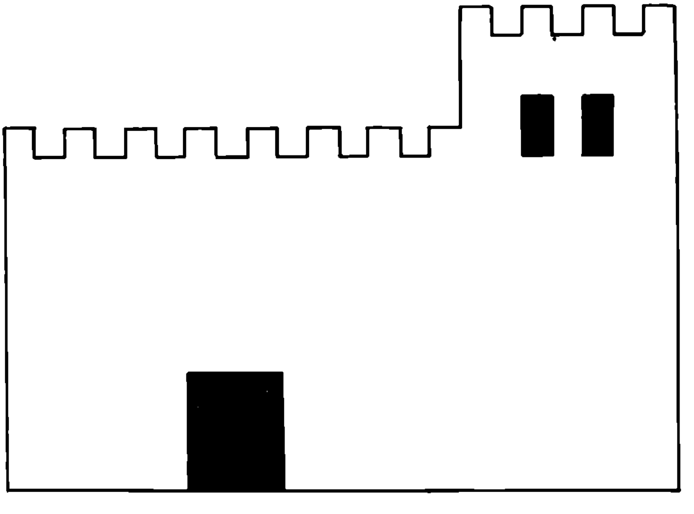
aquí debes poner las subrutinas 200 y 400.

....

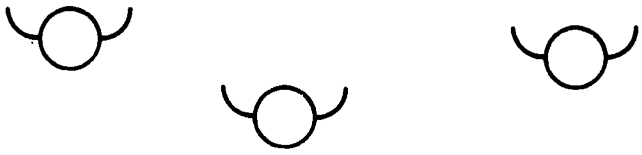
```
610 DATA "[CTRL] [6] [CTRL] [9] ",39,7,15,0:REM
    SUELO
620 DATA "[CTRL] [8] [CTRL] [9] ",14,8,9,8:REM
    PARED
630 DATA "[CTRL] [3] [CTRL] [9] ",14,4,5,8:REM
    TECHO
640 DATA "[CTRL] [1] [CTRL] [9] ",3,5,12,13:REM
    PUERTA
650 DATA "[CTRL] [2] [CTRL] [9] ",2,3,11,9:REM
    VENTANA 1
660 DATA "[CTRL] [2] [CTRL] [9] ",2,3,11,18:REM
    VENTANA 2
670 DATA "[CTRL] [5] [CTRL] [9] ",2,3,3,19:REM
    CHIMENEA
```

EJERCICIOS

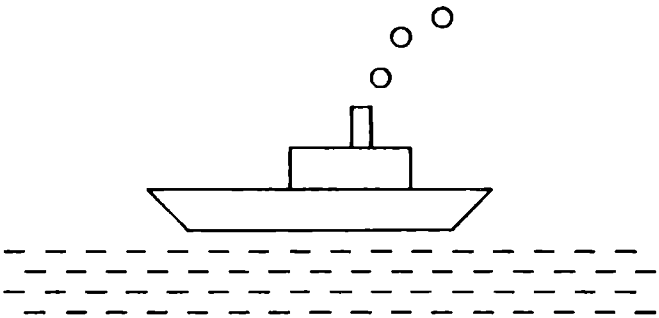
24.1 Haz un programa que pinte esta fortaleza. El suelo ha de ser verde, el cielo azul y las ventanas y puertas oscuras.



24.2 Consigue que estos pájaros muevan las alas.



24.3 Dibuja este barquito en la pantalla. Combinando alternativamente las líneas de rayas podrás conseguir el efecto del oleaje.



24.4 La letra π parece un caballito. Si haces la asignación C$= " π", y luego imprimes C$ muchas veces en la misma línea, tendrás una fila de caballos:

π π π π π π π

Pero si haces C$ = " π" (con un espacio a la izquierda de π), y luego imprimes C$ en la misma línea precedido de TAB(I); (para I = 1, 2, 3, 4...), entonces el espacio borrará el caballito anterior, y tendrás la sensación de que el caballito está corriendo. Hazlo. ¡A ver qué pasa!

25

ON-GOTO

25.1 SALTOS MULTIPLES

Vamos a generalizar las bifurcaciones. Supón que queremos que el ordenador reciba un número, entre uno y cuatro, y lo escriba con letras. Lo podemos hacer con las bifurcaciones normales, así:

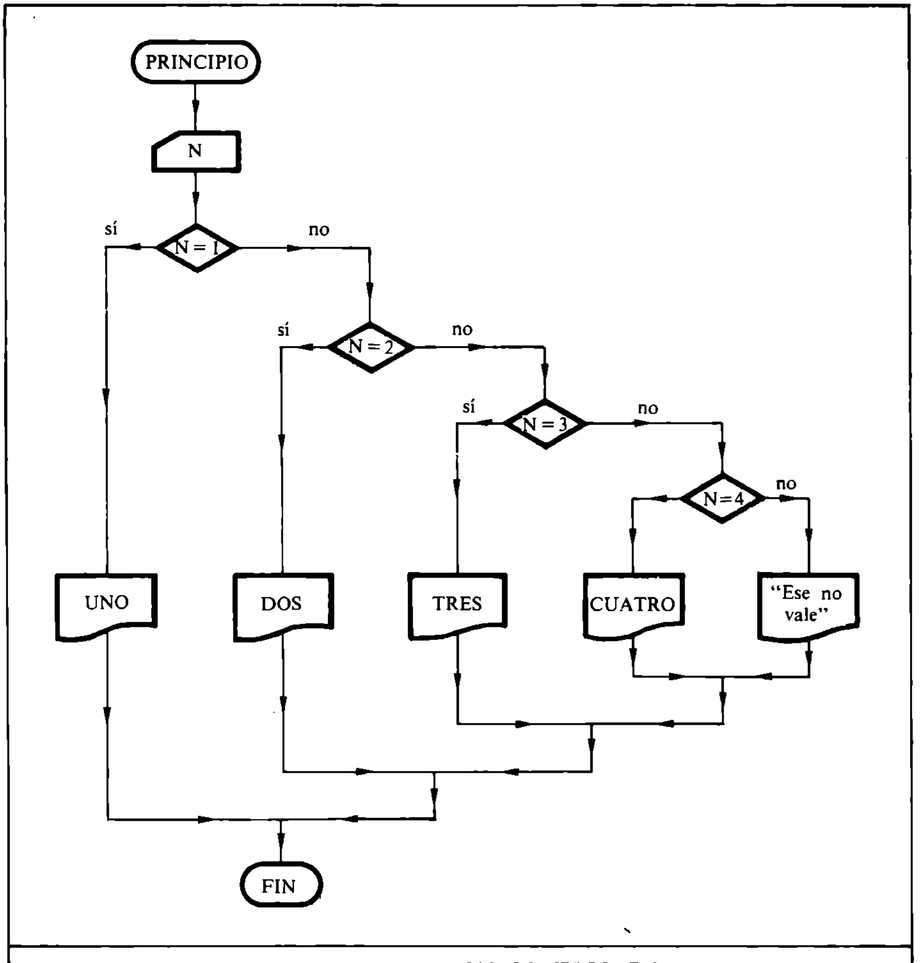


DIAGRAMA DE FLUJO COMPLICADO

Como ves resulta complicado, y no tiene por qué serlo, porque tú lo haces sin ordenador de una manera más fácil, que es como explica este diagrama de flujo:

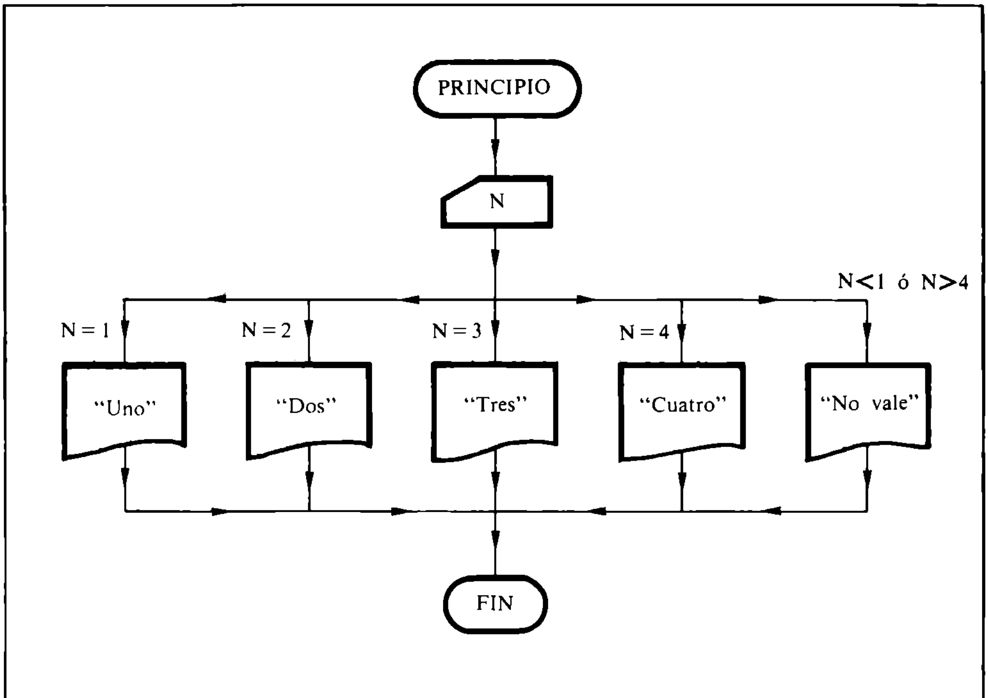


DIAGRAMA DE FLUJO DE NUMEROS HABLADOS

25.2 ON-GOTO

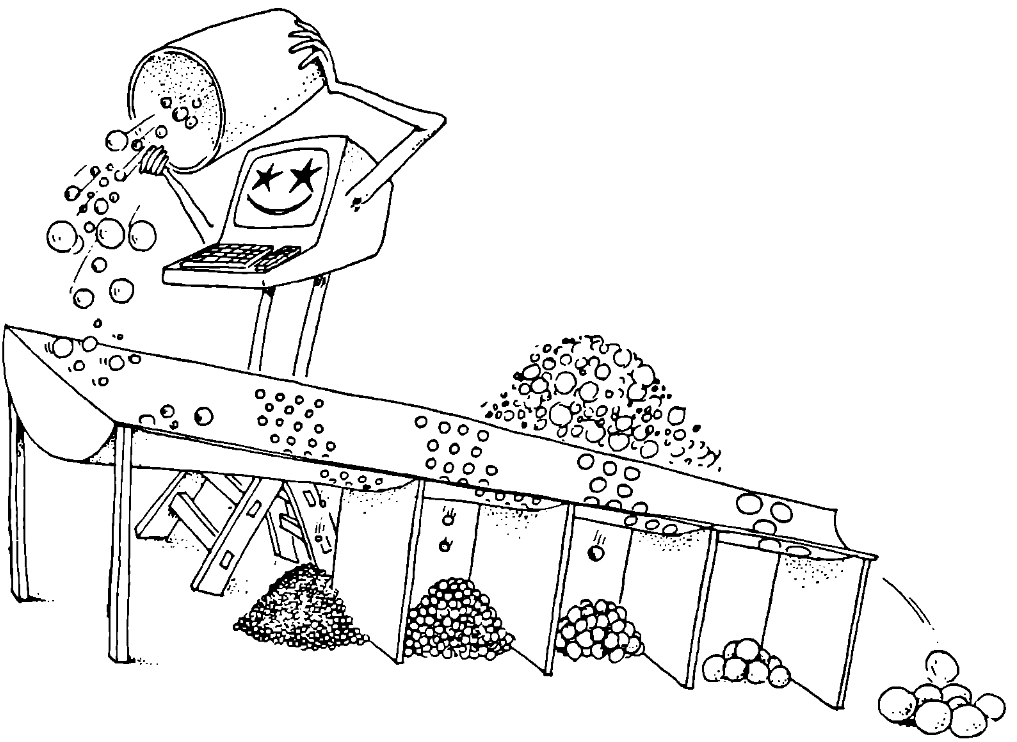
En BASIC también podemos hacerlo así de fácil, porque existe la instrucción ON-GOTO que se usa de la siguiente manera:

```
40 ON N GOTO 100, 200, 280
50 ...
```

Si N vale uno se pasa a la línea 100, si vale dos a la 200, si vale tres a la 280 y si vale menos que uno o más que tres, entonces se continúa en la línea siguiente, la 50.

El programa en BASIC es éste:

```
10 REM NUMEROS HABLADOS
20 INPUT "DAME UN NUMERO";N
30 ON N GOTO 100,200,300,400
40 REM SI N ES MENOR QUE UNO O MAYOR QUE CUATRO
   SIGUE AQUI
50 PRINT "ESE NO VALE"
60 GOTO 1000
100 REM N VALIA 1
110 PRINT "UNO":GOTO 1000
200 REM N VALIA 2
210 PRINT "DOS":GOTO 1000
300 REM N VALIA 3
310 PRINT "TRES":GOTO 1000
400 REM N VALIA 4
410 PRINT "CUATRO"
1000 END
```



Estudiemos sobre este programa cómo se comporta la instrucción ON-GOTO: al llegar el ordenador a la línea 20, comprueba que hay varias direcciones, y si el valor de N es 1, entonces va a la primera dirección que ahí se le indica; si N es 2, va a la segunda; y lo mismo para N igual a 3 y 4. Si N es menor que 1 o mayor que 4, entonces no va a ninguna de estas direcciones, y continúa en la línea siguiente, en este caso en la línea 40.

EJERCICIOS

25.1 La cafetería Micro ofrece a sus clientes los platos combinados 1, 2, 3, y 4. Utiliza la instrucción ON... GOTO... para confeccionar un programa de modo que, al pulsar un número, el ordenador muestre en la pantalla la composición del plato correspondiente.

25.2 Diseña un programa para que aparezca en la pantalla el siguiente "menú":

1. ADIVINANZA
2. REFRAN
3. CHISTE

ELIGE UN NUMERO

A continuación debe aparecer el texto correspondiente al número elegido.

25.3 Tienes un programa con tres juegos. El juego número uno es de marcianitos, el número dos de cartas y el número tres es el dominó. Como eres algo olvidadizo no has puesto la forma de ir directamente a cada juego. ¿Cómo harías el inicio del programa para poder ir al juego deseado?

25.4 Prepara un programa en el que el ordenador te pregunte dos números; a continuación, si quieres sumarlos, restarlos, multiplicarlos o dividirlos; y, según tu contestación, te presente el resultado.

BIBLIOGRAFIA

AGUADO-MUÑOZ, R.; BLANCO, A.; ZABALA, J.; y ZAMARREÑO, R.: *Basic básico. Curso de programación.* Editan los autores. 9ª edición. Madrid, 1984.

AGUADO-MUÑOZ, R.; BLANCO, A.; RUBIALES, E.; ZABALA, J.; y ZAMARREÑO, R.: *Programas comentados de Basic básico.* Editan los autores. 5ª edición. Madrid, 1984.

BLANCO, A., y COMPOSTELA, B.: *El Basic del Spectrum. Del teclado al microdrive.* Editan los autores. Madrid, 1984.

De los mismos autores:

BASIC-BASICO

PROGRAMAS COMENTADOS DE BASIC-BASICO